AD-A054 308 INCO INC MCLEAN VA F/G 9/2
TRANSPARENT INTEGRATED INTELLIGENCE NETWORK - RESPONSE NORMALIZ--ETC(U)
APR 78 P STYGAR F30602-77-C-0030
UNCLASSIFIED INCO/1088-1277-TR-55-D(F) RADC-TR-78-74 NL

1 OF 2
ADA
054308

RADC-TR-78-74
Final Technical Report
April 1978

TRANSPARENT INTEGRATED INTELLIGENCE NETWORK - RESPONSE NORMALIZATION

Paul Stygar

INCO Incorporated

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-78-74 has been reviewed and is approved for publication.

APPROVED: Patricia M Langendorf

PATRICIA M. LANGENDORF
Project Engineer

APPROVED: H Davis

HOWARD DAVIS
Technical Director
Intelligence & Reconnaissance Division

ACCESSION for

NTIS	White Section	☑
DDC	Buff Section	☐
UNANNOUNCED		☐
JUSTIFICATION		

BY
DISTRIBUTION/AVAILABILITY CODES

Dist.	AVAIL. and/or SPECIAL
A	

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (IRDA) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: RADC-TR-78-74	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): TRANSPARENT INTEGRATED INTELLIGENCE NETWORK - RESPONSE NORMALIZATION		5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report. 15 Dec 76 - 14 Dec 77
		6. PERFORMING ORG. REPORT NUMBER: INCO/1088-1277-TR-55-D(F)
7. AUTHOR(s): Paul Stygar		8. CONTRACT OR GRANT NUMBER(s): F30602-77-C-0030
9. PERFORMING ORGANIZATION NAME AND ADDRESS: INCO Incorporated, 7916 Westpark Drive, McLean VA 22101		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62702F, 45941025
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (IRDA), Griffiss AFB NY 13441		12. REPORT DATE: April 1978
		13. NUMBER OF PAGES: 116
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same		15. SECURITY CLASS. (of this report): UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer:
Ms. Patricia Langendorf (IRDA)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Data Base Transparency
Distributed Data Base

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This ~~final~~ report presents preliminary specifications for a subsystem to normalize the query report formats received from heterogeneous DBMS in a Transparent Integrated Intelligence Network (TIIN).

DD FORM 1473, 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

407 275 LB

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (CONTINUED)

TECHNICAL REPORT SUMMARY

The objective of the TIIN/RN project is the design and preliminary specification of the Query Response Normalization subsystem of the Transparent Integrated Intelligence Network. The TIIN system will provide direct analyst access to remote and heterogeneous intelligence data resources without requiring detailed technical familiarity with diverse data management systems. The RN subsystem provides the services required to store, process, and translate data lists prepared as responses to analyst-entered queries.

Specifically, the RN project produced the following technical results:

- Common format specifications for transmission of query response files from diverse host data base management systems. The protocol is called the Query Normal Format (QNF).
- Program specifications for the host RN module that will connect formatted reports into the QNF
- Specification of capabilities required of a common report writer (display generator)
- Specification of a file access interface for application programs to access response file data.

The general methodology employed in performing the design and development tasks was to review and utilize appropriate previous technological accomplishments associated with the TIIN Program, as well as to apply modern information processing concepts. A review of literature on the subject of data translation and data description language was performed; documentation on the DIAOLS, TILE, and SEA WATCH systems was used as source data for design of the QNF and response normalizer module; and, previously developed TIIN design concepts were applied to ensure continuity of the total system development. This effort was performed in parallel with the design of the TIIN Analyst Access Procedures.

With completion of the Response Normalizer Design Specification, the immediate implication for further research is to validate this design as part of the integrated TIIN system design. This can be accomplished through a prototype software model development using an appropriate high order language, thus allowing rapid and inexpensive concept validation.

EVALUATION

This report discusses how to normalize the responses received through accesses to diverse data bases, a necessary aspect of the overall problem of distributed access to non-homogeneous data bases.

Its major contribution will probably be in support of accurate costing of the consequences of non-homogeneity. This will facilitate more cost-effective design choices in the data handling procedures which must be made to implement the delegated production authority.

Patricia M. Langendorf

PATRICIA LANGENDORF
Project Engineer

SECTION I

INTRODUCTION

1. OVERVIEW

This Final Technical Report details the preliminary software specifications of the Response Normalization (RN) subsystem of the Transparent Integrated Intelligence Network (TIIN) as developed under Rome Air Development Center contract number F30602-77-C-0030.

TIIN is intended to provide intelligence analysts with access to remote data base intelligence resources without requiring detailed technical familiarity with the diverse data base management systems, query languages, and data base administration policies at different host nodes in the potential worldwide network of intelligence resources. We recognize that true transparency is a very long-range goal, and that user prompts and user mediation will be necessary in the forseeable future to achieve multiple access to diverse DBMS, in the sense that response to one query defines the next.

A short description of the TIIN system is provided in Appendix B of this document. For a complete account of the background of the TIIN/RN effort the reader is referred to the INCO publications listed in the bibliography in Appendix C.

2. PROJECT OBJECTIVES

The development effort for a TIIN Response Normalization Subsystem, referred to as TIIN/RN, represents INCO's third contract effort concerned with development of a Transparent Integrated Intelligence Network (TIIN). The objectives for the TIIN/RN project are concerned with specification of a common format for transmission of query response files from diverse host data management systems, specification of the host RN module to convert formatted reports to the intermediate response file, specification of a common report writer, and specification of a file access interface for application programs to access response file data.

3. DEVELOPMENT APPROACH

The Response Normalization project was divided into a collection of tasks concerned with research, design, specification and technical reporting. The research tasks were concerned with assessing different host report formats, assessing differences in value encodement schemes, and assessing different techniques for value translation and file conversion.

The design tasks were concerned with the internal format for normalized report file and the interface for application programs to access report files. The specification tasks were concerned with the specification of a report normalization module, a display generator, and an interface for application programs. The technical reporting task was accomplished by preparing technical memoranda to serve as preliminary versions of the material presented in this Technical Report.

a. Report Formats Survey

The first task was to survey the different report formats used in host systems, to determine the specific and the general characteristics necessary for specifying a report normalization module. Sample reports for DIAOLS were obtained through INCO personnel assigned to the NMIC Modernization project at the DIA. For other host systems such as TILE the survey relied on report examples in the TILE User Manual. With most host DBMS it is possible to control the report format to expedite the report normalization process. The DIAOLS format was selected as the candidate for a specification of a report normalizer module.

b. Format for Normalized Reports

Another task was to specify the format for normalized reports. The preliminary specification consisted of a flat file structure, to simplify the design of TIIN modules which process the report file. The flat file structure was rejected as too bulky since line capacity is a critical resource in determining response time. The final specification consisted of a hierarchical file structure with variable length single-valued fields and repeating segments, to allow maximum flexibility in interfacing with a variety of host report formats.

c. Report Normalization Module

A module to perform report normalization was specified for the DIAOLS report format. Several versions were drafted and rejected due to a desire to minimize the effects of random fluctuations in the report format. The version in this report resulted from a top down structured design and was judged relatively simple to implement and maintain.

d. Value Translation Requirements

One task was to determine what techniques are necessary to perform translation of value codes. Several examples of value translation problems were collected and assessed to determine necessary features of a value translation processor.

e. Value Translation Techniques

Several works in the available literature deal with general approaches to file conversion, though the problem of value translation has received little direct attention. The original goal was to develop a declarative language for users to use to specify value translations, as part of the data description capability for the Network Access Directory. However, the requirements for value translation were judged to be too complex for a declarative language. The full capabilities of an existing procedural language are required.

f. Application Program Interface

The interface for application programs to access normalized report files is essentially an interface to access a hierarchical file. A precise specification of the interface was not attempted since it is dependent on the selection of a data base management system (DBMS) to perform data management services for the TIIN system.

g. Display Generator

The module to reformat a normalized report file was kept deliberately simple. Its unique feature is the presentation of units of measurement with each field. Presumably after a TIIN DBMS is selected the Display Generator module will be adapted or embedded in the report formatter of the DBMS.

4. CONCLUSIONS AND FUTURE DEVELOPMENT

It has been a basic theme in the TIIN concept that TIIN is essentially a virtual DBMS, that is, TIIN itself provides no data management services, since all data production is performed at the host sites aligned with the data sources. The user query language (TEL) is essentially a virtual query language, since it retrieves data only through conversion into another query language which is in turn processed by a DBMS to extract data from a production data base. However, with the response normalization study there has been a growing realization that many of the capabilities of TIIN are basically DBMS capabilities, for example:

- o After a query report has been normalized the optimum format for the normalized report is some type of self-descriptive file
- o To view the response file there must be a capability for report generation

- o After value translation the response file will tend to be in an unsorted state, so a sort capability is needed

- o To perform analysis and/or complex value translation the capability required is an interface to user written applications programs, preferably an interface which takes advantage of the self-descriptive nature of the response file

- o The information in the TIIN Network Access Directory is basically a collection of hierarchical files, so a file access utility would be appropriate.

a. Choice of Descriptor Driven DBMS

The alternatives for future development of TIIN focus on a few key decisions. Perhaps the most important decision is whether to use an existing DBMS or to develop a descriptor-driven DBMS to perform the data management services required in the TIIN system. This decision is important since use of an existing DBMS would shorten the time for actual implementation of TIIN, would avoid the problems of developing "yet another" DBMS, and would permit future TIIN development efforts to concentrate on breaking new ground.

The choice of a DBMS, however, is not a simple decision if it is based entirely on developmental considerations. Perhaps a better criteria would be minimal impact on the existing collection of analysis programs, since it is very difficult to redevelop analysis programs to operate under a new DBMS, particularly in the intelligence community where access to programs must be strictly controlled.

b. Analyst Requirement for Standardization

One basic issue, addressed in the study effort, is the provision of a capability for translating between the different data encodement schemes which exist in the separate files in the potential network of data bases. This problem is not intrinsic to the intelligence community. It arises because data base designers must balance so many interdependent options that the results often seem random even when the designers all have the same data sources, the same DBMS, the same constraints and the same community of prospective users. The problem of heterogeneous

data encodement schemes is the result of the natural tendency towards diversity which accompanies the development of any new industry due to the need to experiment with different techniques to seek better local optimums. The producers exert a centrifugal force, the tendency towards diversity, which limits the utility and the potential of the industrial raw material, in this case, information. The consumers exert a centripetal force, the tendency towards uniformity, which seeks to minize recurring production costs and to maximize recurring benefits. In the long term the solution to the problem of heterogeneous data encodement will be to adopt and enforce data encodement standards during the data collection process itself. The adoption of standards is a centripetal movement which must be based on consumer experience with actual data usage. The purpose of the DBMS network is to provide a medium in which consumers can access data from different producers and subsequently perform analysis and derive benefits. The DBMS network must provide software tools which support data conversion to allow users to gain experience with data standards.

c. Analyst Requirement for Data Distribution

In a network of distributed data bases there is a tendency for an analyst to replicate data. A few production data bases stimulate a vast amount of reproduced data. Though there is no formal "Law of Conservation of Information," it is intuitively obvious that a DBMS cannot improve the quality of the data after the data has been produced and distributed. That is, it is not feasible to update every copy of a particular data value when that value has been determined to be erroneous or inaccurate. To some extent it may be possible to provide a capability for systematic notification of updates similar to the message distribution capability currently provided by some host systems. The problem of data distribution is a general problem in a network of data bases, and will continue to be an appropriate area of investigation. The alternative to an organized approach to data distribution in an integrated network is that the network will be saturated by analysts who find it necessary to reproduce their local data base on a regular basis.

5. REPORT ORGANIZATION

The following paragraphs summarize the five major sections and the appendices of this report. The appendices contain significant information and should be considered as part of the main body of the report. A reader who is not familiar with the TIIN effort might begin with Appendices A and B.

a. Section I: Introduction

The first section provides an overview of this Final Report in fulfillment of the contract and in relation to the overall TIIN development.

b. Section II: Report Normalization

The second section provides a short survey of report formats, a detailed description of the DIAOLS report format, the specification for a module to perform report normalization for the DIAOLS report format.

c. Section III: Value Translation Interface

The third section provides a short survey of value translation problems and a description of which problems can be handled transparently in query normalization and report normalization.

d. Section IV: Display Generator

The fourth section defines the user interface and the program specifications for the display generator, which provides the user the capability to reformat a normalized report.

e. Section V: Application Program Interface

The fifth section provides a description of the function and preliminary design for an interface to provide a capability for application program access to normalized report files.

f. Appendix A: Development Background

The development background describes the software environment and the transparency concepts underlying the TIIN system.

g. Appendix B: Description of the TIIN System

The TIIN System consists of five subsystems with modules and module interfaces.

h. Appendix C: Bibliography

The bibliography lists the research material relevant to the TIIN Response Normalization project.

SECTION II

REPORT NORMALIZATION

1. OVERVIEW OF REPORT NORMALIZER MODULE

The function of the report normalizer is to insure that query responses from heterogenous DBMS are received in a common format at user nodes in the TIIN system. The functional purpose of the common format is to provide a uniform convention for data management services to the analyst via the TIIN system.

What the report normalizer module does is convert a host-to-user report into the TIIN internal format for a hierarchical data file. The reason it is necessary to perform data input to the network by converting host-to-user reports is that this approach has no impact on the host software: that is, the network is transparent to the host DBMS and the host operating system.

The following subsections provide descriptions of the following:

- o Short survey of possible host report formats
- o Specification of Response Normal Format
- o Specification of a Report Normalizer Module.

Throughout this section most of the examples of reports and report normalization will be based on the query report format generated by the DIAOLS host DBMS.

Please note that the Response Normal Format (RNF) like the Query Normal Format (QNF) is not intended to be any type of stand-alone communications protocol. All the information shown is assumed to be sent throughout the network in the WICS Common Format. This format will take care of the delivery of the Response File to the appropriate node. The information contained within the Response File is strictly for use by the TIIN software. For a description of the WICS Common Format, see the Standard Software Base Final Technical Report dated 15 September 1976. On page III-7 of the report the outgoing traffic format is laid out. The Response Normal File is assumed to be contained in the message area.

The Report Normalizer is part of the TIIN Response Normalization Subsystem. Other components perform functions as listed below:

- o Value conversion from host-to-network or from network-to-user or from host-to-user frame of reference
- o Display generation to present the normalized report in a user-oriented format.

2. REPORT FORMATS SUMMARY

A formatted report received from a DBMS is equivalent to a hierarchically structured file: the first sort key in the report corresponds to the key item of a level one segment, while secondary sort keys in the report correspond to dependent segments. The basic difference between a formatted report and a hierarchical file is that a report contains a high percentage of overhead information in comparison with the physical structure used to store a hierarchical file on a mass memory device. The overhead information in a report is designed to facilitate human access to the data, whereas the overhead information in a hierarchical file is designed to facilitate programmed access.

There are endless varieties of report formats. In most cases there is an element name or title associated with each value. The association is usually achieved by positioning the name or title to the left of or above the value. When the association is horizontal an equal sign (=) may serve to separate the name from the value.

a. Vertical Association

A vertical association between a label and a value is achieved by positioning the label above the value. The simplest example of this type of association occurs in a columnar report, as shown in Figure II-1.

Label 1	Label 2	. . .	Label N
Value 11	Value 21	. . .	Value N1
Value 12	Value 22	. . .	Value N2
			.
			.
Value 1M	Value 2M	. . .	Value NM

Figure II-1. Label and Value Vertical Association

(1) Example From DIAOLS

As illustrated in Figure II-2, the DIAOLS system will automatically produce a columnar report in response to a query if there are less than ten columns. Note that a column label may be columnized, i.e., split across two or more lines (e.g., "SHORT NAME" is printed with "SHORT" on one line and "NAME" on a second line immediately below "SHORT").

(2) Example From TILE

As illustrated in Figure II-3, the TILE system has the capability to specify the order and column position for elements retrieved via a query.

```
                              UNCLASSIFIED
                                                            76/08/12   PAGE 1
SHORT      LONG/SYNONYM
NAME           NAME           SIZE   TYPE     FORMAT
0001*      DATE-OF-CHANGE     008     F     ALPHANUMERIC
0010$*     RECORD-ID          009    I-M    ALPHANUMERIC
0020       TITLE              025     V     ALPHANUMERIC
0100       TEXT-1             065     V     ALPHANUMERIC
0110       TEXT-2             065     V     ALPHANUMERIC
0120       TEXT-3             065     V     ALPHANUMERIC
0130       TEXT-4             065     V     ALPHANUMERIC
0140       TEXT-5             065     V     ALPHANUMERIC
0150       TEXT-6             065     V     ALPHANUMERIC
0160       TEXT-7             065     V     ALPHANUMERIC
0170       TEXT-8             065     V     ALPHANUMERIC
0180       TEXT-9             065     V     ALPHANUMERIC
0190       TEXT10             065     V     ALPHANUMERIC
```

Figure II-2. Example of DIAOLS Record Display

```
     EXTI/CAR;MAKE(BUICK*)
     AND COST<(26ØØ).
     PRINT/CAR;
     OPTION:HEADER, SPACE(1)
     FORMAT:PART
     MAKE(1-25), COST(27-35), BODY(37-44), GEAR(46-5Ø)

THE FOLLOWING ANSWERS WERE DERIVED AT Ø745     EST    ON JULY 21,7Ø

FROM QUERY NUMBER     312171ØØØ1

TOTAL RECORDS EXTRACTED=       3

MAKE                         COST             BODY           GEAR

BUICK SPCL STANDARD V8       2414             2 DR SDN       STAND
BUICK SPCL STANDARD V8       2468             4 DR SDN       STAND
BUICK SPCL STANDARD V8       2561             4 DR SDN       STAND


END OF RUN
```

Figure II-3. Example From TILE

b. Horizontal Association

A horizontal association between a label and a value is achieved by positioning the label to the left of the value, as illustrated in Figures II-4a and II-4b.

```
Label 1 = Value 11
Label 2 = Value 21
        ...
Label N = Value N1

Label 1 = Value 12
Label 2 = Value 22
        ...
Label N = Value N2
        .
        .
        .
Label 1 = Value 1M
Label 2 = Value 2M
        ...
Label N = Value NM
```

Figure II-4a. Label and Value Horizontal Association - Columnar

```
Label 1 = Value 11, Label 2 = Value 21, ..., Label N = Value N1.

Label 1 = Value 12, Label 2 = Value 22, ..., Label N = Value N2.
        .
        .
        .
Label 1 = Value 1M, Label 2 = Value 2M, ..., Label N = Value NM.
```

Figure II-4b. Label and Value Horizontal Association

(1) Example 1 From DIAOLS

The DIAOLS system will automatically use a non-columnar report if the columns side-by-side would take more characters than the teletype line width permits. Note the problems illustrated in the record display in Figure II-5: (1) page break occurs in the middle of a record; (2) indices for "relational periodic elements" need not match, as seen in the second record; (3) label and value may be displayed in left half page or in right half page depending on length of value, as seen in the display for "REMARKS-1".

```
RECORD-ID         = GFØ7                DATE-OF-CHANGE      = 76Ø821
NAME              = PEARL HARBOR        LATITUDE            = 2125Ø6N
LONGITUDE         = 1575Ø22W            CATEGORY            = 12895
COUNTRY-CODE      = US                  STATUS              = INACTIVE
USAGE             = AIRFIELD
DATE-OF-MISSION   = (Ø1) 66Ø3Ø7
                    (Ø2) 67Ø228
                    (Ø3) 681Ø1Ø
MISSION-NUMBER    = (Ø1) 42567XXX
                    (Ø2) 62419MXM
                    (Ø3) 47622MMM
LOCATION          = PACIFIC
REMARKS-1         = SEAPLANE STATION
REMARKS-2         = STRIKE AIRCRAFT SEEN
RECORD-ID         = GFØ8                DATE-OF-CHANGE      = 76Ø821
NAME              = SUBIC BAY           LATITUDE            = 1445Ø1N
LONGITUDE         = 12Ø1311E            CATEGORY            = 12895
COUNTRY-CODE      = PI                  STATUS              = INACTIVE
USAGE             = INACTIVE
DATE-OF-MISSION   = (Ø1) 66Ø923
                    (Ø2) 67Ø831
MISSION-NUMBER    = (Ø1) 42631XXQ
                    (Ø3) 45961XXX
NAVAL-SHIPS       = RIVER-ATTACK-CFT

                          UNCLASSIFIED


                          UNCLASSIFIED
                                              76/Ø8/3Ø  PAGE 2

LOCATION          = PACIFIC             REMARKS-1           = NOT COMPLETED
REMARKS-2         = ALL SHIPS DEPARTED
```

Figure II-5. Example of DIAOLS Record Display

(2) Example 2 From DIAOLS

The DIAOLS system will continue a value onto a second line, indented, with the value break occurring on a blank, as illustrated in Figure II-6.

```
RECORD-ID          = ISSCURRØ7          DATE-OF-CHANGE   = 76Ø415

TITLE              = USER NUMBER LOCKOUTS

TEXT-1             = NEW SECURITY PROCEDURE, USERS BEWARE---- AS OF 19 APRIL

     76 USER

TEXT-2             = NUMBERS WILL BE LOCKED OUT OF HELLOONE (I.E. THE USER

     WILL BE

TEXT-3             = DISCONNECTED AND WILL NOT BE ABLE TO SIGN BACK ON THE

     SYSTEM).

TEXT-4             = THIS WILL APPLY TO THE FOLLOWING CONDITIONS -

TEXT-5             = 1.  SIGNING ON TO 2 TERMINALS SIMULTANEOUSLY IN ISS WITH

     1 USERID.

TEXT-6             = 2.  FAILURE TO INPUT CORRECT PASSWORD - SECOND ATTEMPT.

                    UNCLASSIFIED                                 S45Ø57

                    UNCLASSIFIED

                                                        76/Ø7/Ø9  PAGE  2

TEXT-7             = 3.  FAILURE TO INPUT VALID ACTION - SECOND ATTEMPT.

TEXT-8             = 4.  FAILURE TO INPUT VALID CLASSIFICATION (MFQ-II) -2ND

     ATTEMPT.

TEXT-9             = IF YOUR USER # GETS LOCKED, CALL SO-6 (MR MCCORMACK)

     26334/26384.

TEXT1Ø             = ***********************************************************

     *********
```

Figure II-6. Example of DIAOLS Record Display

(3) Syntax of DIAOLS Report

From manual examination of actual host reports it is possible to infer the syntax of a DIAOLS report. A summary of the possible line types is presented in Figure II-7.

Some lines (null line) contain no query response data, only classification or page numbers or host system messages. Some lines (full line) have one label and one value. Some lines (contline) represent continuation of the value from the previous full line. Some lines consist of two columns (leftcol, rightcol) each of which can contain a label value pair (elval or relperiodic) or an extra value (multival or relmultival) for the label in the same column immediately above the extra value.

The specification of the report normalizer involves making numerous unverifiable assumptions as to the composition of the incoming report format. The analysis of the format presupposes the existence of common text delimiters such as LF/CR, equal (=) signs, tab and its associated skip counter, a maximum characters/line length, and an EOF (End of File).

There are two (2) major line format types:

o label value, and

o free format text composition.

The label value type is composed of label descriptors with no embedded spaces and unique values. This type is columnar. Some horizontal print lines may have two (2) labels and two (2) values, while others may have but one label and one value per line. Periodic elements are additionally included. These may be relational or non-relational. Each periodic value following the first is on a separate line and contains no preceding equal (=) sign or label. Relational periodics contain an index in parentheses.

The free format text type is of mixed composition - columnar and text lines. The columnar type appears as two (2) distinct sub-types - discrete values and embedded text lines. The discrete values are similar to those of the label descriptor type. The embedded text line occurs with two (2) text line number labels per line, and may or may not contain any value data. Additionally, there are full text lines with no value data and full text lines with special characters used as record dividers. The text line type is most notably distinguished by its indented continuation data line(s), although a text line may only be one (1) line in length. That is, text lines are constructed in two (2) ways - block format and indented format. In block format each line is defined by

line	nulline	
	leftcol, rightcol	
	fullline	label, tab (18), '=', value, tab (75)
	contline	tab (6), value, tab (75)
nullline	blankline	tab (75)
	unclassified	tab (21), 'UNCLASSIFIED', tab (75)
	confidential	tab (21), 'CONFIDENTIAL', tab (75)
	secret	tab (21), 'SECRET', tab (75)
	topsecret	tab (21), 'TOP SECRET', tab (75)
	pagenum	tab (55), digits, '/', digits, '/', digits, 'PAGE', digits, tab (75)
leftcol	leftnull	tab (39)
	leftelval	label, tab (18), '=', value, tab (39)
	leftrelperiodic	label, tab (18), '= (', digits, ')', value, tab (39)
	leftrelmultival	tab (20), '(', digits,')', value, tab (39)
	leftmultival	tab (20), value, tab (39)
right	rightnull	tab (75)
	rightelval	tab (39), label, tab (57), '=', value, tab (75)
	rightrelperiodic	tab (39), label, tab (57), '= (', digits, ')', value, tab (75)
	rightrelmultival	tab (59), '(', digits,')', value, tab (75)
	rightmultival	tab (59), value, tab (75)

Figure II-7. Line Types in a DIAOLS Query Report

(Notation: tab(n) means blanks up to column 'n' after preceding symbol.)
Refer to Figure II-11 for the syntax-directed actions corresponding to the above syntax description.

a label and follows an equal (=) sign. In the indented format a defined text line value is continued on the following print line. The indented line begins in character position six (6), (left margin), and may not even end a sentence, but appear as one (1) word within a sentence that is continued on the following line with a normally positioned label descriptor. It is not known whether or not multiple indented text lines exist, or if these contain no words, one word, or a data line. From a review of actual reports there appears to be only one indented line allowed for each label descriptor.

There is also a third type of report format encountered which appears as the least unmixed of all types. This type is columnar, but may contain two (2) label descriptors on some lines. Periodic elements appear to be absent. This format is similar to the label-descriptor value type, and is mentioned here to show the heterogenity of these reports.

Periodic elements pose a particularly difficult problem in that they may appear anywhere within the report in either column. It is assumed that multiple values for a label will always appear in the same column as the label. The locations of both periodic and non-periodic elements as they might appear in a host formatted report are shown in Figure II-7.

It was assumed that character/code locations, i.e., equal sign (=), multi-labels per line, are fixed, and, therefore, report analysis could be distinct and sharp. This appears not to be the case as no standard column positions can be established for all report formats. The necessity to define a universal format scanner thus becomes of primary importance. Assumptions must be made as to each format's internal composition and common distinguishing features. With insufficient information as to how the report exists (upon receipt) and the dependence on unverified, delimiting criteria, some concretizing of the report formats is necessary.

c. Segmented Associations

A collection of label-value associations is called a segment. A segment of one type can be associated with one or more segments of another type. Usually the one occurs above the many, perhaps as a page header or as a header for a set of columns.

3. RESPONSE NORMAL FORMAT

The functional purpose of a common format for transmission of data files on the network is to provide a uniform convention for data services to the analyst at the TIIN user node. Several advantages result from this function, as follows:

- o Data communication costs are reduced if the conversion occurs before the file is transmitted across the communication lines.

- o The information in a file in common format may be processed by application programs much more readily than information in a variety of different formats.

- o The information in a file may be processed to translate value codes from the host frame of reference to the user frame of reference.

a. Design Decisions

The design of the common format for response files, called Response Normal Format (RNF), incorporates the following decisions:

- o All fields are represented as variable length (a character string preceded by a count) rather than as fixed length to permit maximum flexibility in a system such as TIIN which interfaces with many different types of DBMS.

- o Records are represented as a hierarchical structure of repeating segments since this is more compact than a flat file and corresponds directly with the structure of host reports and response files.

- o The response file includes its own description, so the file can be processed by utility routines independently of its method of origin.

b. Description of RNF

The Response Normal Format (RNF) as shown in Figure II-8, consists of a header and a sequence of data records. The header identifies and completely describes the response file of which it is a part. The TIIN/QDNC subsystem will use this information for data stream merging as well as routing to the final destination node. The other information contained in the header is the number of records in the data file, the number of types in the data file and the origin of the file (i.e., the node within the network where the file was built). The number of unique segments and the maximum depth of any segment are contained in the header.

Data records consist of a series of segments. The file header has segment descriptors which show the interrelationships between

segments and the composition of each segment. Each segment descriptor has a unique identification number and a sequence of field descriptors for the data values which the subsequent data records contain. Each field descriptor indicates the field name, the unit-of-measurements, and the maximum value length.

Each record consists of a set of ordered segments which in turn consist of an ordered list of data values. The ordering of the data values in a segment of a data record correpponds one-for-one with the ordering of field descriptors shown in the segment descriptor. The ordering of segments within a record is identical to the order obtained by traversing the segment tree, representing the record, in preorder. That is, each dependent segment occurs after its parent, and a parent segment does not occur until after all occurrences of dependent segments of the preceding parent segment. The number of segment descriptors is limited to 255.

An RNF response file consists of a header and a series of data records all of which have the description contained in the header.

HEADER	RECORD 1	RECORD 2	...	RECORD N

The header consists of a file descriptor and a series of segment descriptors.

FILE DESC	SEG DESC 1	SEG DESC 2	...	SEG DESC K

Each record consists of a record header and a series of data segments.

REC INFO	SEGMENT 1	SEGMENT 2	...	SEGMENT L

Each segment descriptor consists of a series of field descriptors.

SEG ID	FIELD DESC 1	FIELD DESC 2	...	FIELD DESC M

Each segment in a data record consists of a series of data values. When the identified (ID) of a segment descriptor matches the identifier of a data segment, then the data values of the record segment correspond one-for-one with the field descriptors of the segment descriptor.

SEG ID	VALUE 1	VALUE 2	...	VALUE M

Figure II-8. Layout of RNF

c. Contents of RNF

The network standard data format is shown in diagram form in Figure II-9. A more detailed description of each field is given below.

(1) Header

The header's function is to identify the data stream which follows. An approximation of the size of the file is also included.

(a) Data File Indicator - Specifies that the header which follows is a data file header (i.e., as opposed to an error message header, status header, QIP transmission, etc.)

(b) Query Identifier - Ties the following data file to a specific user query.

(c) File Identifier - Maps the following data file to a specific branch in the operator tree.

(d) Originating Node Identifier - Specifies the node from which this data file was originated.

(e) Number of Segment Descriptors - Specifies the number of segment descriptors which describe the data file and immediately follow the header.

(f) Maximum Segment Nesting Depth - Specifies the largest segment nesting level. The root node of the operator tree is level 1 and each succeeding level increases the nesting depth by 1.

(g) Number of Records - The number of data records contained in this particular partial data stream. For DIAOLS this number is known. For SEAWATCH and TILE it may have to be an estimate.

(h) Number of Bytes - An estimate of the size of the response file. Calculated by multiplying the number of records by the average number of types per record. This will require a NAD entry for each segment specifying the average number of bytes per segment type per record.

(2) Segment Descriptor

There is one segment descriptor for each segment in the record including the root segment.

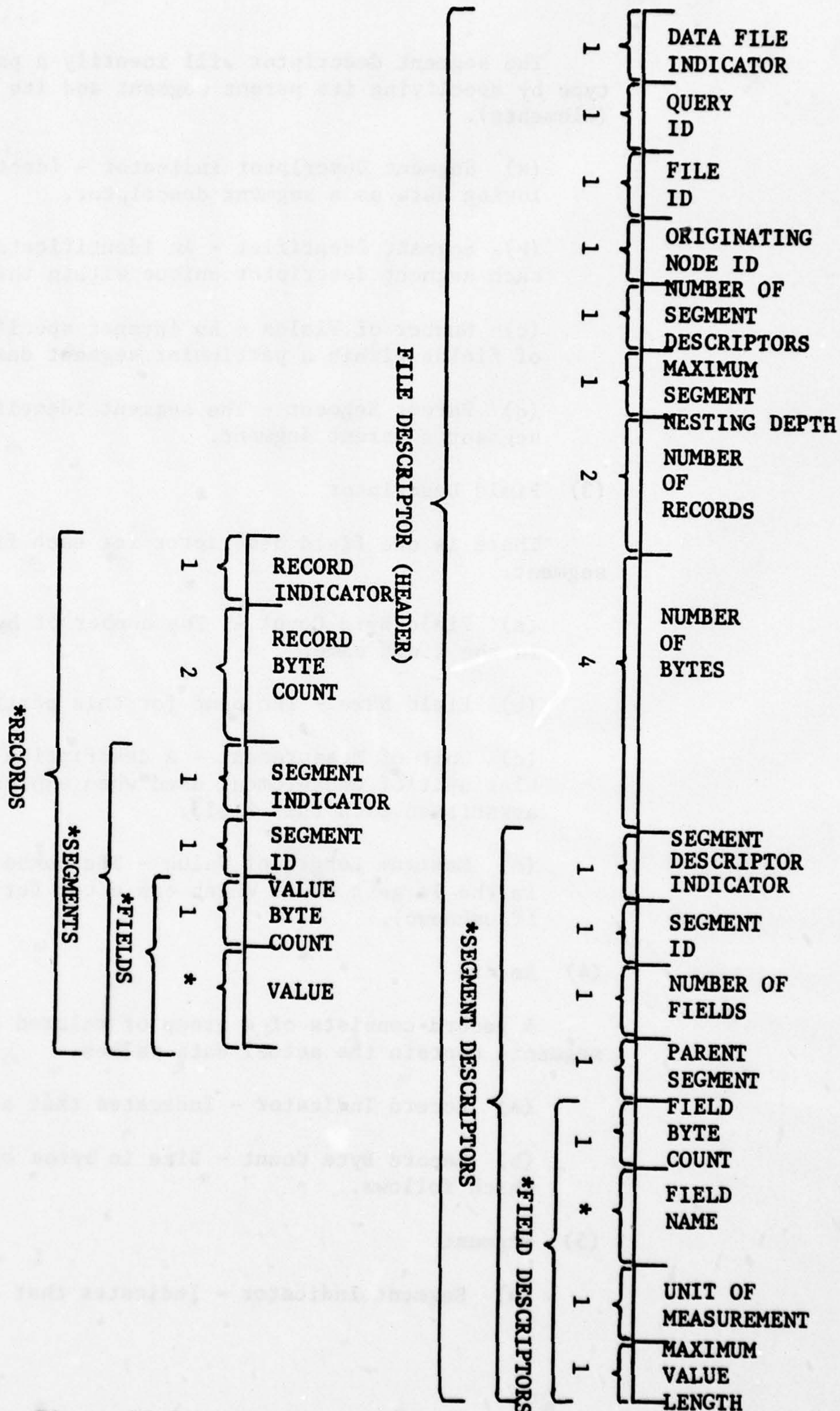


Figure II-9. Contents of RNF

The segment descriptor will identify a particular segment type by specifying its parent segment and its component fields (elements).

(a) Segment Descriptor Indicator - Identifies the following data as a segment descriptor.

(b) Segment Identifier - An identification number making each segment descriptor unique within this file.

(c) Number of Fields - An integer specifying the number of fields within a particular segment descriptor.

(d) Parent Segment - The segment identifier of this segment's parent segment.

(3) Field Descriptor

There is one field descriptor for each field in the segment.

(a) Field Byte Count - The number of bytes contained in the field name.

(b) Field Name - The name for this particular field.

(c) Unit of Measurement - A description of the particular unit of measurement used when expressing values associated with this field.

(d) Maximum Length of Value - The number of characters in the largest value which can occur for this field (Ø if unknown).

(4) Record

A record consists of a group of related segments. The segments contain the actual data values.

(a) Record Indicator - Indicates that a record follows.

(b) Record Byte Count - Size in bytes of the record which follows.

(5) Segment

(a) Segment Indicator - Indicates that a segment follows.

(b) Segment Identifier - Maps each segment to the segment descriptor which describes it (i.e., they will have identical segment identifiers.)

(6) Field

(a) Value Byte Count - Specifies the length of the value which immediately follows.

(b) Value - A data value in its network standard representation.

The items which are considered to be special cases along with their meanings are as follows:

- o A value byte count of zero represents the null value.
- o A record number of zero represents the end of file record.
- o A segment descriptor with a parent segment equal to zero is the root descriptor for the hierarchical segment tree.

4. GENERAL APPROACH FOR REPORT NORMALIZER

The Report Normalizer (RN) receives the following inputs:

- o From the Query (the QNF of the TIIN/QIP subsystem) the RN receives a list of the requested file elements to be included in the report.
- o From the NAD the RN receives for each element the type, maximum size, title, segment identifier, and host-intrinsic role (periodic, relational periodic).
- o From the Host DBMS the RN receives the unnormalized report.

A report normalizer module should use the structural information about the data elements described in the Host NAD. The query specifies which data elements are retrieved from the Host Data Base for inclusion in the report. For these retrieved elements the report normalization module can obtain the following information from the Host NAD.

Element name
Size
Format
Structural relation (parent)

This information from the Host NAD must be used to form the descriptor for the response file. The file descriptor consists of segment descriptors which consist of value descriptors. The file descriptor is transmitted before the actual data records so that a response file is self-descriptive.

The report normalizer module uses the file descriptor as a skeleton from which to build each data record. As the report is scanned the various elements are recognized, delimited validated, and sorted in the record skeleton, as follows:

recognize	- obtain element name, then find slot in skeleton
delimit	- obtain element value from report
validate	- compare value with size and format expected
sort	- the position of the value indicates the element name and the value relationship to elements with the same name.

The Report Normalizer consists of the following functional aspects:

- o File Element List processing which combines the QNF and NAD information to produce an RNF file header including segment descriptors with field descriptors
- o Report line recognition which classifies each line of the report into one or more of the finite set of line syntaxes characteristic of the particular host
- o RNF generation which outputs a record after all the field values have been assembled into segments.

The overall goal for design of a report normalizer module is to have a simple organization with a collection of utility routines. The utility routines should be general purpose enough to be usable in report normalization modules for other hosts.

a. Co-ordination Between Host QIP and Host RN

The Host QIP module converts the QNF version of the query into the Host Language version. It has control over which elements are to be retrieved by the Host DBMS for inclusion in the Host report. For some hosts the Host Language may permit a great deal of control over the composition of the host report, as follows:

- o It may be possible to instruct the host DBMS to produce a fixed format file instead of a formatted report, which has obvious efficiency advantages.

- It may be possible to specify substitutes for the element names in the report, to simplify the problem of element name recognition.

- It may be possible to specify a vertical listing of element name and value pairs, which would simplify the scanning procedure.

- It may be possible to specify the order in which elements are displayed, to eliminate the sorting problem by using the same element order for both the host report and the TIIN response file.

For the DIAOLS host DBMS none of these efficiencies is feasible. For the TILE host DBMS it is possible to specify column positions in a vertical listing.

b. Technical Problems in Report Normalization

Several technical problems must be resolved to permit reliable modularized conversion of formatted reports into normalized response files.

- Host OS messages and error bursts may occur unpredictably. This requires an error recovery mechanism to remove the effect of unpredicted messages, and to resume processing after encountering an error, since a teletype interface does not provide a capability to request retransmission of erroneous data. Some messages occur repeatedly without change and can easily be filtered out.

- Differences in report styles for different DBMS may be significant enough to proclude a generalized modular design for a report normalization module.

- There are many potential syntactic ambiguities since the same delimiters (i.e., blanks, equal signs, parentheses) may be used for field names and values.

- Blank values are sometimes not printed.

The problem of detecting errors and recovering after errors is too difficult to attempt a complete solution. The problem should be

gnored since transmission errors cannot be the responsibility of the user node. One possible error recovery mechanism would be to abandon the current record and scan the report until a new record is encountered. However, this would lose a great deal of data potentially useful to the user. Perhaps the abandoned record should be flagged as erroneous and transmitted as completely as practical with the response file.

5. REPORT NORMALIZER FOR DIAOLS

The processing summary for the TIIN Report Normalizer module for DIAOLS is given in Figure II-11, indicating the syntax-directed actions corresponding to the syntax description of the DIAOLS query report format given in Figure II-7. Essentially these two figures provide a structured specification of the Report Normalizer module. A flow diagram specification is provided in Figures II-13 through II-21. Though a host query report contains much that is ambiguous from a free format point of view, the use of column position information provides a feasible basis for avoiding many ambiguities, and the general technique of syntax-directed processing appears to be a reasonable approach for this data processing problem.

a. Summary of Report Normalizer Subroutines

The overall program flow of the Report Normalizer (RN) is shown in Figure II-13. The subroutines are described briefly below:

- o RN MAIN - The main routine serves as a driver only, it calls the initialization routine (FEL INIT) and the report scanner (LINE SCAN).

- o FEL INIT - The initialization routine uses the query and the NAD data descriptions to build the RN internal table (called the File Element List or FEL) and the RNF header consisting of a file descriptor, segment descriptors and field descriptors.

- o LINE SCAN - The host report is processed line by line with syntax recognition of the labels and data values in each line, with values stored in a FIFO list of value packets associated with each label in the FEL.

- o RN ERROR - Process error conditions detected by subroutines in the RN module.

- o RNF GEN - The list of segments is obtained from the FEL and used to call the subroutines which output the values for each segment.

- o VALPROC - The host segment is produced from the nonperiodic elements (i.e., the single valued elements in the FEL).

- o PERECON - A segment instance is produced for each value packet of each simple periodic element in the FEL.

- o PESAS - The value packet lists of each set of related relational periodic elements are used to produce a sequence of RNF segments.

(1) Design Goals

There are several distinct levels of host report processing required in the RN module. The complexity of design in this software component of TIIN lies in the desire to achieve generality by avoiding intrinsic characteristics of the DIAOLS report format. Though the DIAOLS report format illustrates most of the problems that might be encountered in other host DBMS, it has not been feasible to use the DIAOLS report format as the basis for a fully generalized TIIN Report Normalizer. The design of the report normalizer has been improved and generalized in several design iterations, though the amount of generality achieved cannot be measured.

The DIAOLS record structure is at best a two-level hierarchy, with one root segment and all dependent segments strictly immediate descendants of the root segment. Thus the DIAOLS host report format does not serve to illustrate the hierarchical aspects of report normalization, and these aspects have not been included in the specifications for the DIAOLS report normalizer.

(2) Design Assumptions

To achieve generality it is necessary to make as few intrinsic assumptions as possible, though to achieve a finite design and an efficient program it is necessary to make as many specific assumptions as possible. The tradeoff between generality (minimize assumptions) and efficiency (maximize assumptions) has resulted in the following specific assumptions about the DIAOLS report format.

- o The element labels appear unordered on the report.

- o All element labels and values for any one record will appear contiguous, i.e., labels and values associated

with different records do not appear intermixed in the report.

- o Only those element labels and values requested by the query will appear in the host report.

- o Multiple values for a periodic element always appear in the same column vertically contiguous.

- o Equal sign (=) is a reserved delimiter which separates label and value. Equal signs do not occur embedded within value strings.

- o A sequence of two or more blanks is a (tab) delimiter for a label or a value, or a value fragment (if followed by a continuation line).

- o An EOF (end of file) or equivalent occurs at the end of the report to signal the end of Report Normalization processing.

- o Element labels do not have embedded blanks.

- o Any sequence of one or more blanks in a value can be reduced to a single blank if the value is textual.

(3) No Imbedded Equal Signs

The prime key to effective searching of the known host report formats is the equal sign (=) placed between the label and the value.

Although it is possible to invalidate this search technique by allowing embedded equal signs to appear within the body of the text lines (both single and multi-line), no actual embedded equal signs have been found in a review of sample host reports. Therefore, a decision has been made to bypass reference to this condition and this is to treat all found equal signs as value indicators.

Examples of this embedded equal sign condition is shown below in Figure II-10. A reasonable alternative to disallowing embedded equal signs to use column positions is to resolve the ambiguous cases, since the delimiter usage occurs predominately in columns 18 and 57.

TITLE	=	PERIODICS	=	RELATIONAL DATA

Figure II-10a. Single Line Label-Value Pair With Equal Sign Embedded in Textual Value

TEXT-1	=	ISS PERIODIC DATA HANDLING	=	
PROBLEMS	=	COMMENCING EARLY MARCH		

Figure II-10b. Double Line Label-Value Pair with Multiple Equal Signs Embedded in Textual Value

TEXT-1	=	PERIODICS	TEXT-2	=	RELATIONAL

Figure II-10c. Single Line with Left and Right Label-Value Pairs Without Embedded Equal Signs.

b. Report Normalizer Main (RN Main)

The main routine (see Figure II-14) initializes the internal tables for the primary scan and then invokes the primary scan.

c. File Element List Processor (FELINIT)

The purpose of the FEL Processor is to combine the data descriptions in the Host NAD and the list of requested elements in the query (QNF) to produce (1) the internal table required by the Report Normalizer and (2) the header for the Response Normal Format (RNF) to be transmitted with the normalized records. The internal

table is called the File Element List and its purpose is to specify the host attributes for the elements requested in the query. The FEL is sorted by segment identifier, so the RNF header and the RNF records can be generated by a one-pass scan of the FEL. Each FEL entry has a FIFO list of value packets, so the RNF record generator can output field values in the same order as the field descriptors in the RNF header.

Figure II-15 illustrates the logic flow of the File Element List (FEL) processor. Figure II-12 specifies the contents of the File Element List (FEL).

d. Report Line Recognition (Search and Format)

This subroutine (LINESCAN) of the RN module identifies the host element labels in the host report, accesses the attributes of the host element in the File Element List (FEL), creates a FIFO list of value packets, and calls the RNF generation subroutine when the end of record is detected.

The main search is accomplished through a sequential scan of the host report in an identify-and-delimit process. The lines of the report are scanned one by one in turn.

A value packet is constructed for each value string obtained from the host report. The value packet consists of a line (possibly null) to another value packet, plus a byte count and the element value itself, as illustrated below.

LINK	SIZE	VALUE

The processing of each report line is summarized in Figure II-11 by indicating the action performed for each line type, in analogy with the operation of syntax-directed processors. The processing flow is specified as a logic flow diagram in Figure II-15.

e. Error Processor (RN Error)

Various flowchart pointers have indicated the presence of potential error conditions within the RNF effort. Many more can be identified with more knowledge as to the system constraints.

It is sufficient to note that the error condition is being recognized. However, its solution will remain unattempted herein because more information is needed on error types, frequency and procedures to be followed in error conditions.

STATE	SUBSTATE	PRECEDING STATE	ACTION
line	null line	any	ignore. No change in "preceded by" parameter
	leftcol, rightcol	(see below)	(see below)
	fullline	any	verify labels, verify MAX length
	contline	fullline	append value (contline) to value (fullline), verify MAX length
leftcol	leftnull	any leftcol	no action. If rightnull then treat as nulline.
	leftpair	any	verify label, verify MAX length
	leftrelpair	any	verify that label is relational periodic, verify MAX length, store index and value.
	leftrelvalue	leftrelpair	verify MAX length, store index and value with label, obtain label from preceding leftrelperiodic
	leftvalue	leftpair	obtain label from preceding leftpair, verify that label is periodic, verify MAX length
rightcol	rightnull	any	no action. If leftnull then treat as nulline
	rightpair	any	verify label, verify MAX length.
	rightrelpair	any	verify that label is relational periodic, verify MAX length, store index and value.
	rightrelvalue	rightrelpair	obtain label from preceding rightrelperiodic, verify MAX length, store index and value with label
	rightvalue	rightpair	obtain label from preceding rightpair verify that label is periodic, verify MAX length.

Figure II-11. Report Normalizer Processing for Discrete Line Types (Refer to Figure II-7 for the syntax descriptions corresponding to the above syntax-directed actions.)

HOST ELEMENT NAME (STRING)
HOST ELEMENT NAME SIZE (INTEGER)
HOST ELEMENT TYPE (INTEGER)
HOST ELEMENT UNIT OF MEASUREMENT (INTEGER)
HOST ELEMENT MAX VALUE LENGTH (INTEGER)
SEGMENT IDENTIFIER (INTEGER)
PARENT SEGMENT IDENTIFIER (INTEGER)
SPECIAL CHARACTERISTICS (STRING)
FIFO LIST OF VALUE PACKETS (POINTER)
TAIL OF LIST OF VALUE PACKETS (POINTER)

Figure II-12. Contents of File Element List (FEL).

Functionally the line recognizer routine must obtain element labels from the report line, identify the label in the File Element List, obtain the element label from the report, and attach the value to the FEL entry for the label.

During this stage of processing there is no distinction made between simple periodics and relational periodics. The periodic index which appears in parentheses prefixed to each relational periodic value is stored unchanged in the value packet. The index is separated from the base value during the subsequent RNF generation phase.

Error type codes are transmitted to the Error Processor (RN ERROR) and a table look-up match performed to identify the error procedures to follow. These procedures are left undefined for this development effort.

f. RNF Generation (ALIGN)

This routine of the RNF module is called after all label-value pairs for a single record have been collected into the FEL from the host report. The entries in the FEL correspond one-to-one with (and in the same order as) the field descriptors in the RNF header. Each entry in the FEL has a FIFO list of value packets. Hence the value packets in the FEL are in essentially the same order as the field values in the RNF. The primary task of the ALIGN routine is to convert the lists of value packets into a sequence of segments, each segment consisting of a sequence of field values with at most one value per field.

The operation of the RNF Generator is to loop over the different segment identifiers. For each segment identifier a subroutine is called to output all segments with that specific identifier. Three different types of level (or root) segment, (2) the single value segment corresponding to each value of a periodic element, and (3) the segment corresponding to a set of related relational periodic elements. Since DIAOLS handles no more than two levels of record hierarchy, all non-root segments are immediate descendants of the root segment, and all instances of each segment can be generated contiguously.

g. Value Processor for Root Segment

This routine of the RN module is called to output the RNF segment corresponding to the non-periodic elements obtained from the host report. This routine operates by processing each entry in the File Element List (FEL) beginning with the first entry. Since all non-periodic elements occur in the FEL before all the periodic elements, the routine is completed when the first periodic element is encountered in the FEL.

h. Value Processor for Simple Periodic Elements

This routine of the RN module is called to output the RNF segment corresponding to the simple periodic elements obtained from the host report. Simple periodic elements are multi-value elements, as are relational periodic elements, but the separate values are not associated with the values of any other multi-value element. In the RNF representation there is no mechanism for representing multi-value elements per se. Rather, each multi-value

element corresponds to multiple occurrences of a segment which consists of a single element, so that every RNF element is single-valued. In the host NAD every simple periodic element has been assigned a separate segment identifier.

The functional purpose of this routine is to convert a multi-valued host element into a multi-occurring single valued RNF element. This is accomplished by "popping" the FIFO list of value packets and outputting one segment for each value packet.

i. Value Processor for Relational Periodic Elements

This routine of the RN module is called to output the RNF segments corresponding to relational periodic elements obtained from the host report. Two or more relational periodic elements are considered to be interrelated if they have the same segment identifier, and the corresponding elements are combined in common segments in this routine. (Note that the decision to assign two or more relational periodic elements the same segment identifier is decided by the host data base administrator who places data descriptions in the host NAD. The DIAOLS host DBMS does not provide any mechanism for relational periodic elements to be inter-related except via the query using the RELATE verb, and the host DBMS itself is not aware of any implicit relationships between relational periodic elements.)

After the LINESCAN routine has completed its scan, the relational periodic element values are queued up in FIFO lists under each label in the FEL. Each entry in the FIFO list has a relational periodic value consisting of a periodic index and a raw value, formatted as follows:

(NN) VVVVV

The periodic index is an integer from 01 to 99 and indicates the segment instance for the periodic value.

For example, consider an FEL with the following entries:

FIFO LIST OF VALUE PACKETS

ELEMENT NAME	SEGID	LINK	SIZE	VALUE
PORT	2		12	(Ø1) NEW YORK
		Ø	13	(Ø2) BALTIMORE
ARRIVAL-DATE	2		6	(Ø1) 76Ø7Ø1
		Ø	6	(Ø2) 76Ø711
DEPARTURE-DATE	2		6	(Ø1) 76Ø7Ø9
		Ø	6	(Ø2) 76Ø714

This FEL would result in two segment instances, as follows:

$	2	12	NEW YORK	6	760701	6	760609

$	2	13	BALTIMORE	6	760611	6	760614

In order to clarify the function of the translation algorithms and the design of the network standard data format the following examples are presented. They are not intended to be all inclusive but rather to illustrate several of the more obscure points concerning the reformatting of records into a network standard.

Some fields within a record will contain multiple values with identical subscripts. When this occurs RN will be required to generate at least one segment for each value of the subscript. Suppose a record contains the following:

DATE-OF-MISSION	=	(01) 661027
		(02) 670915
		(03) 681010
MISSION-NUMBER	=	(01) 46732XXX
		(01) 46732MXM
		(02) 97283MMM
		(03) 47632MXM
MANPOWER	=	(01) 137
		(01) 89
		(02) 125
		(03) 162

The segment descriptor will contain 3 fields: DATE-OF-MISSION, MISSION-NUMBER and MANPOWER. Since the 1st date of mission corresponds to two mission numbers and to two manpower counts, two record segments will be generated containing the 1st date of mission. In order to do this it must be known that DATE-OF-MISSION, MISSION-NUMBER and MANPOWER are all periodic elements directly relating to each other. This relationship will be shown in the NAD where the three elements have been assigned the same segment identifier by the host data base administrator. The four segments generated will group the data elements as follows:

segment (DATE-OF-MISSION, MISSION-NUMBER, MANPOWER)

661027,46732XXX,137

661027,46732MXM,89

670915,97283MMM,125

681010,57632MXM,162

Thus the original relationship as shown by the subscripts is maintained in the standard format.

It is interesting to note the results if the three elements in the above example had been grouped differently into segments by the host DBA. If DATE-OF-MISSION and MISSION-NUMBER have the same segment identifier while MANPOWER has a different segment identifier the result will be eight segments, four for each of two different segment identifiers, as follows:

segment (DATE-OF-MISSION, MISSION-NUMBER):

661027,46732XXX

661027,46732MXM

670915,97283MMM

681010,57632MXM

segment (MANPOWER)

137

89

125

162

Similarly, if all three elements had different segment identifiers, then eleven segments would result, with three for the segment containing DATE-OF-MISSION, four for MISSION-NUMBER, and four for MANPOWER.

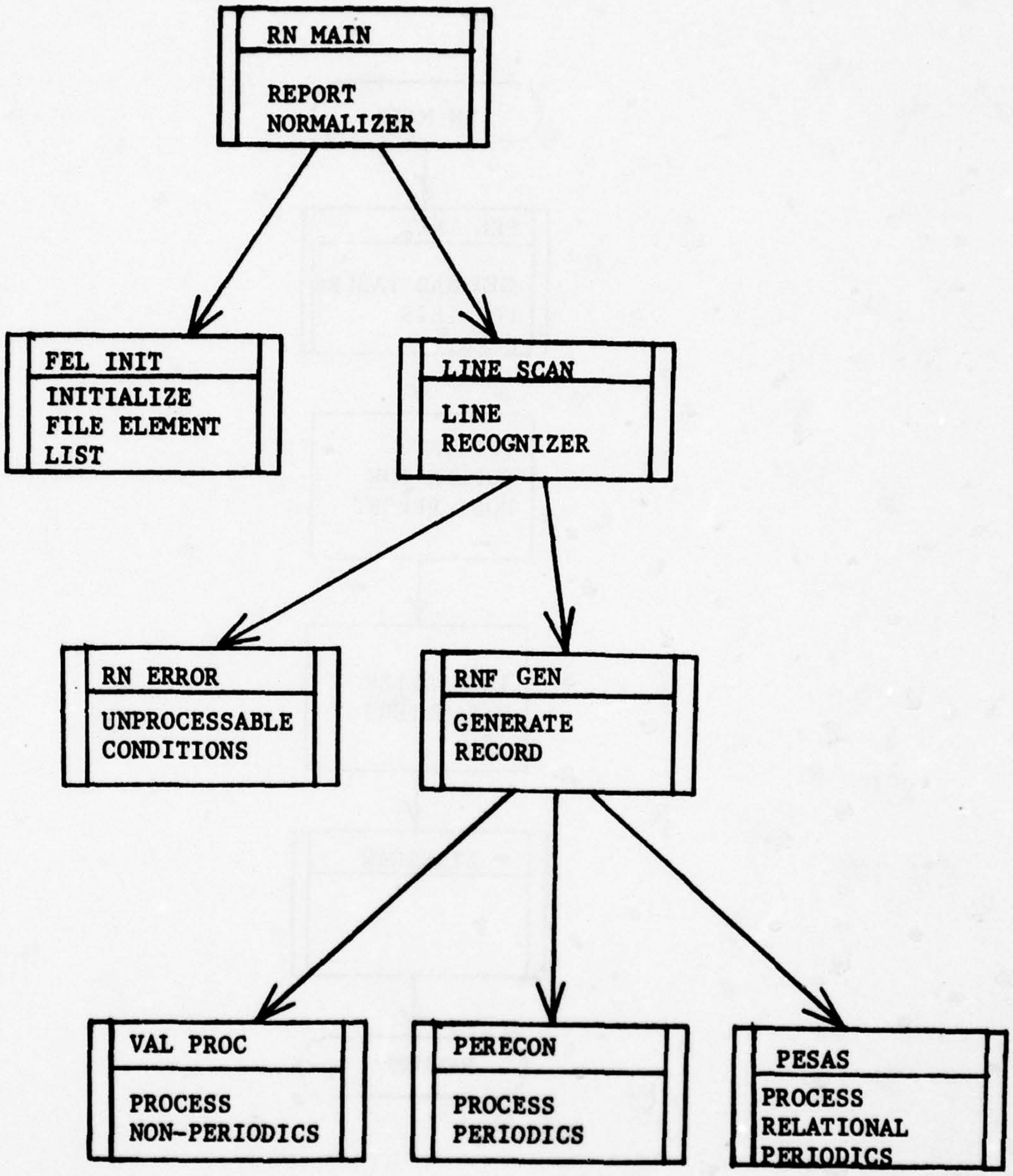


Figure II-13. Logic Flow Between Subroutines in the Report Normalizer (RN) Module.

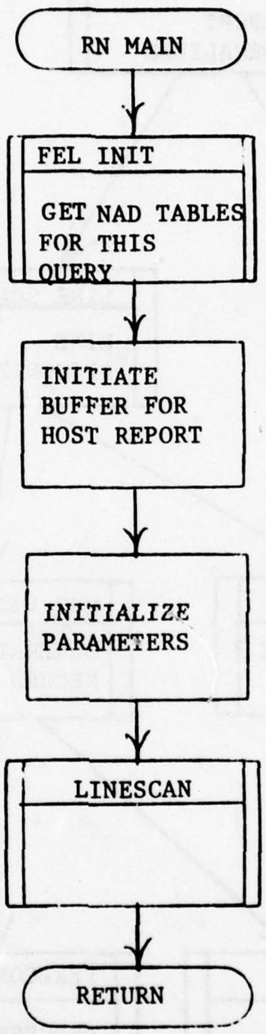


Figure II-14. Flow Diagram for Report Normalizer MAIN Routine.

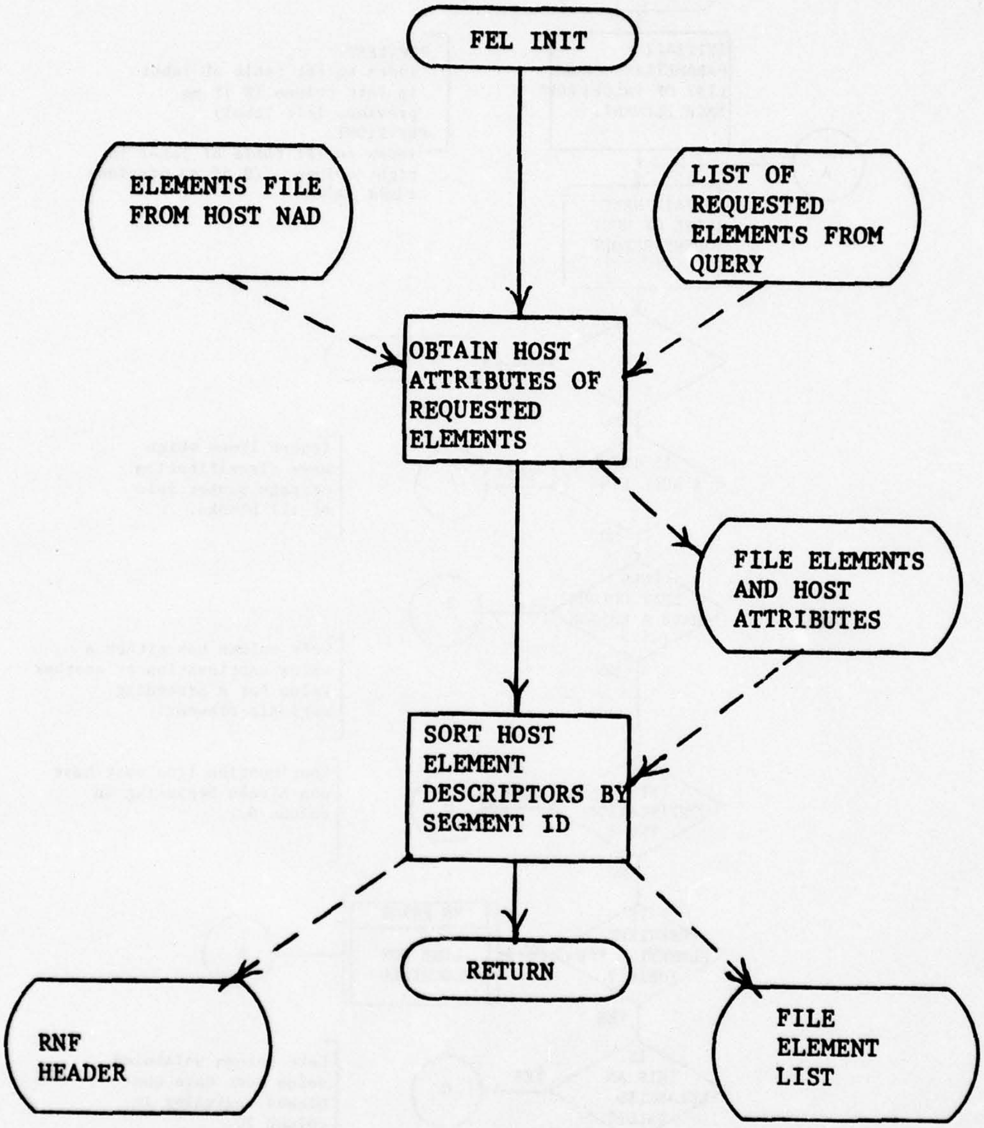


Figure II-15. Logic Flow for File Element List (FEL) Processor.

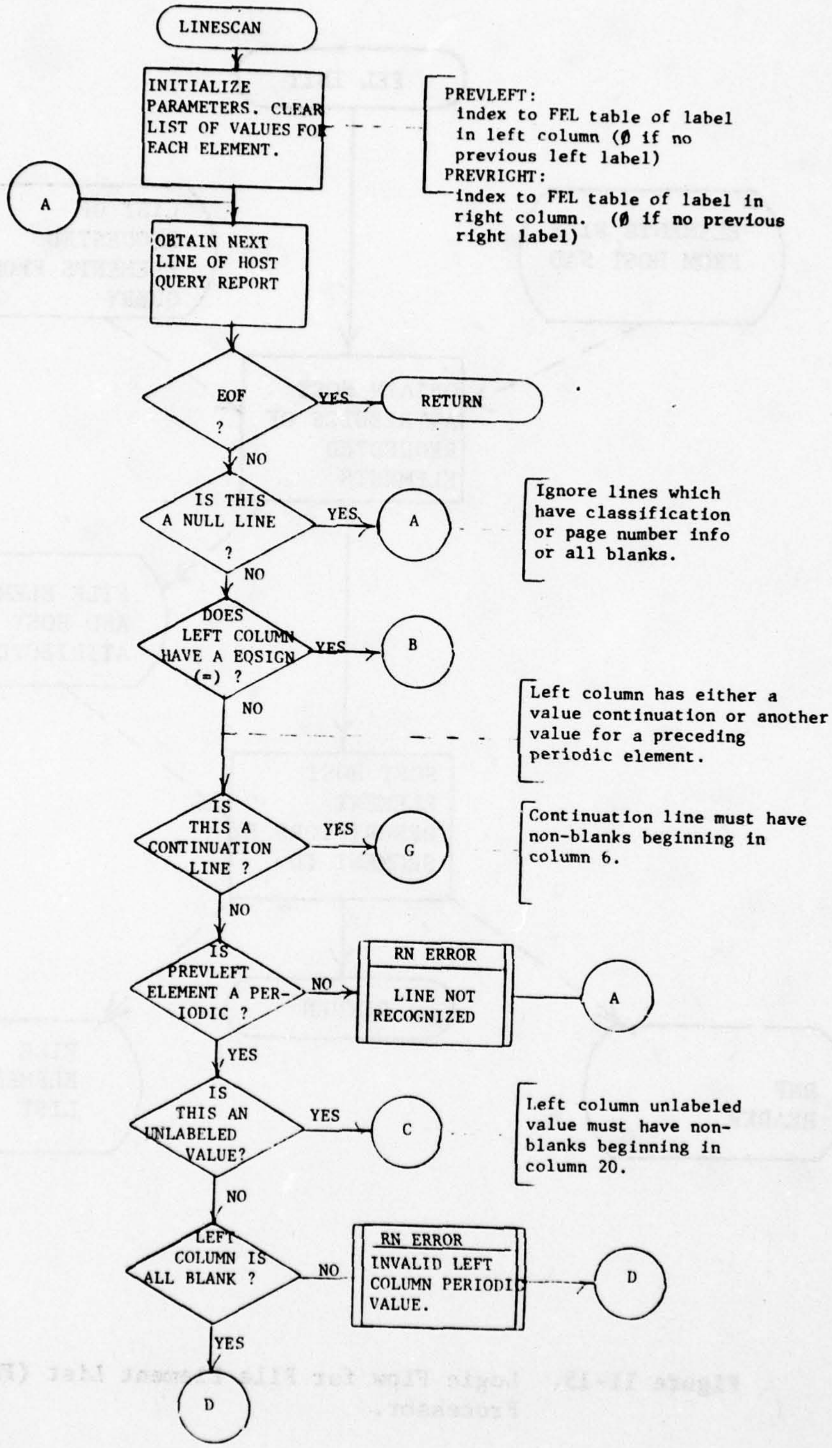


Figure II-16. Flow Diagram of Line Recognizer

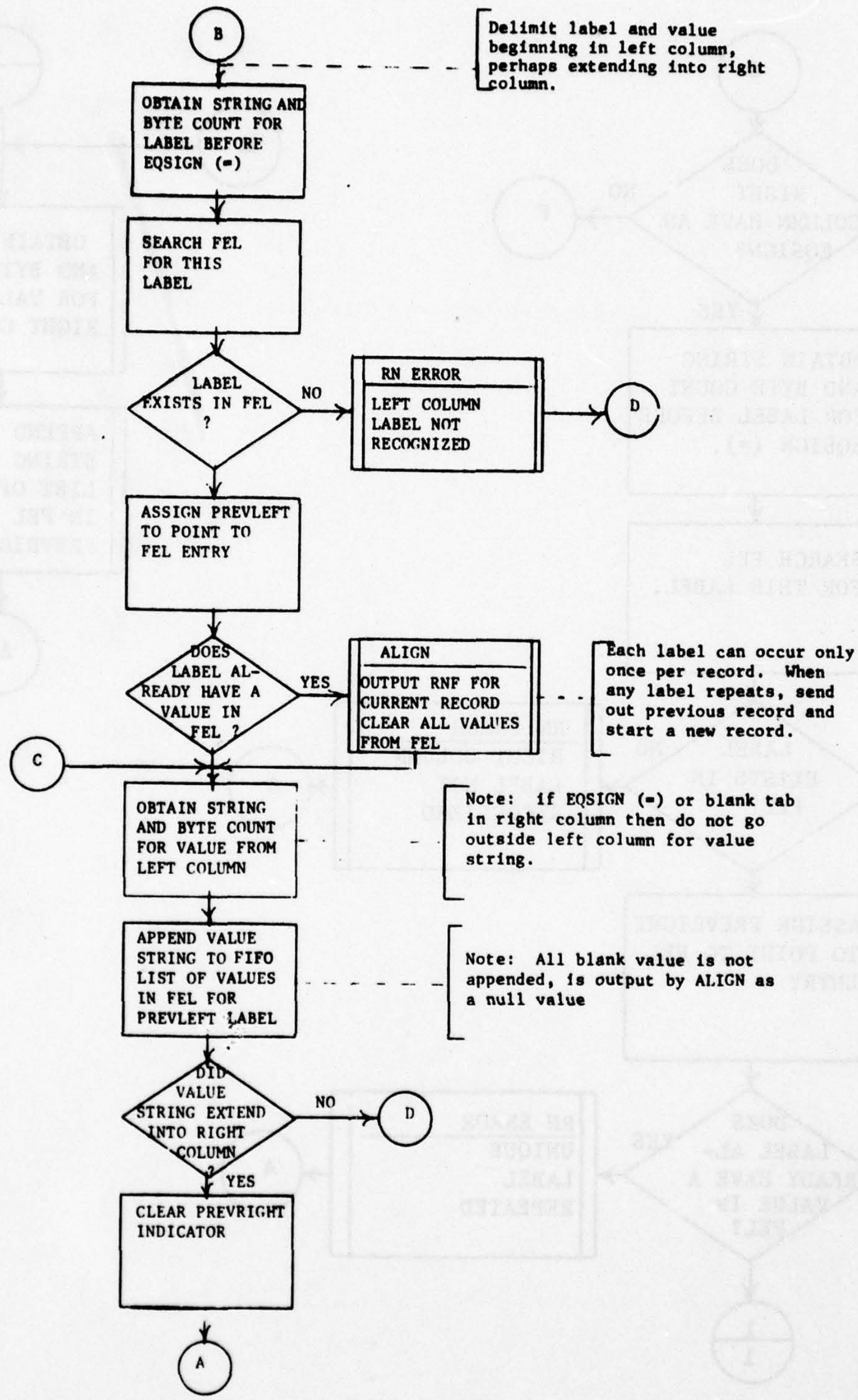


Figure II-16 (Cont.). Flow Diagram of Line Recognizer

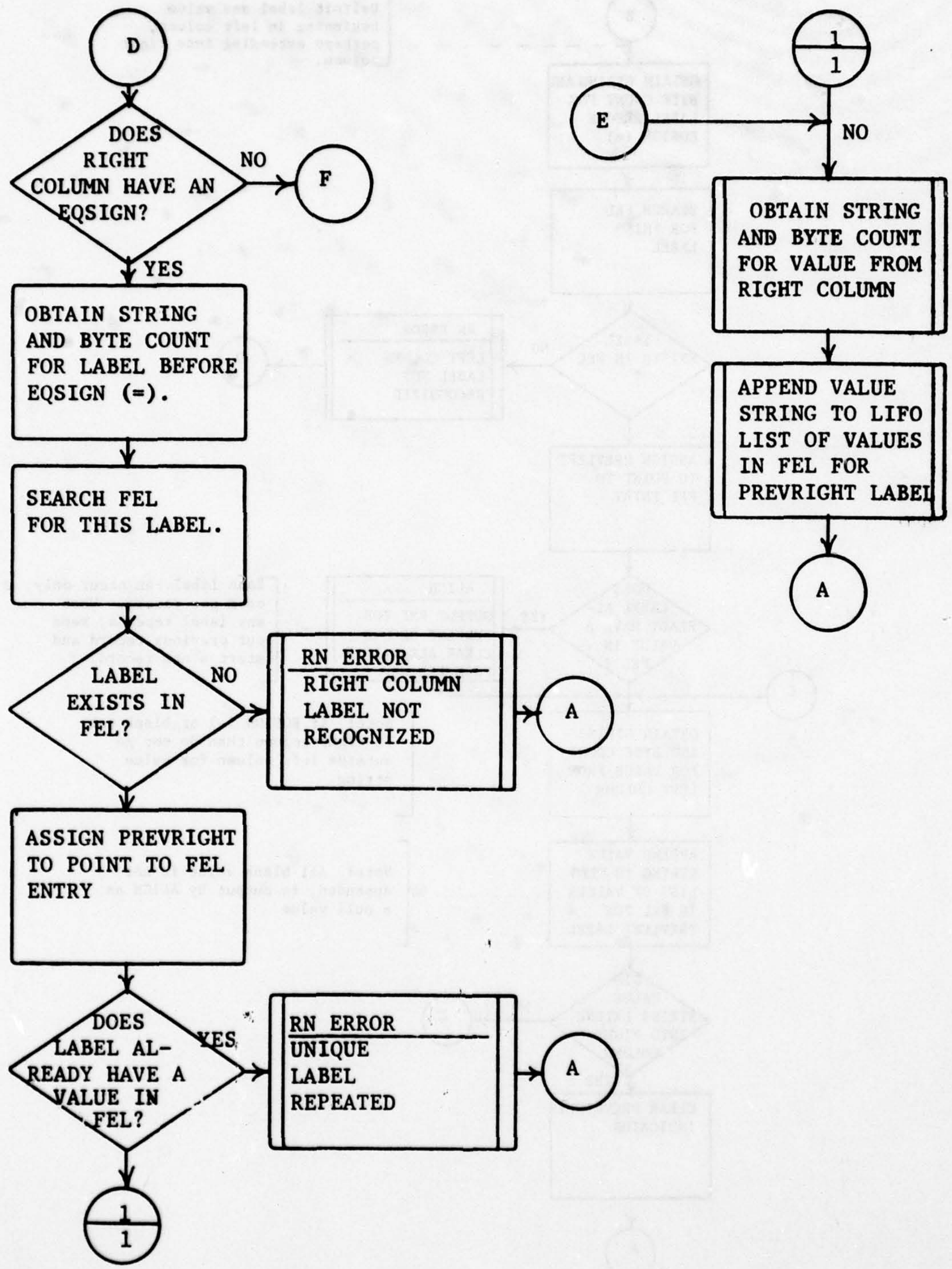


Figure II-16 (Cont.). Flow Diagram of Line Recognizer

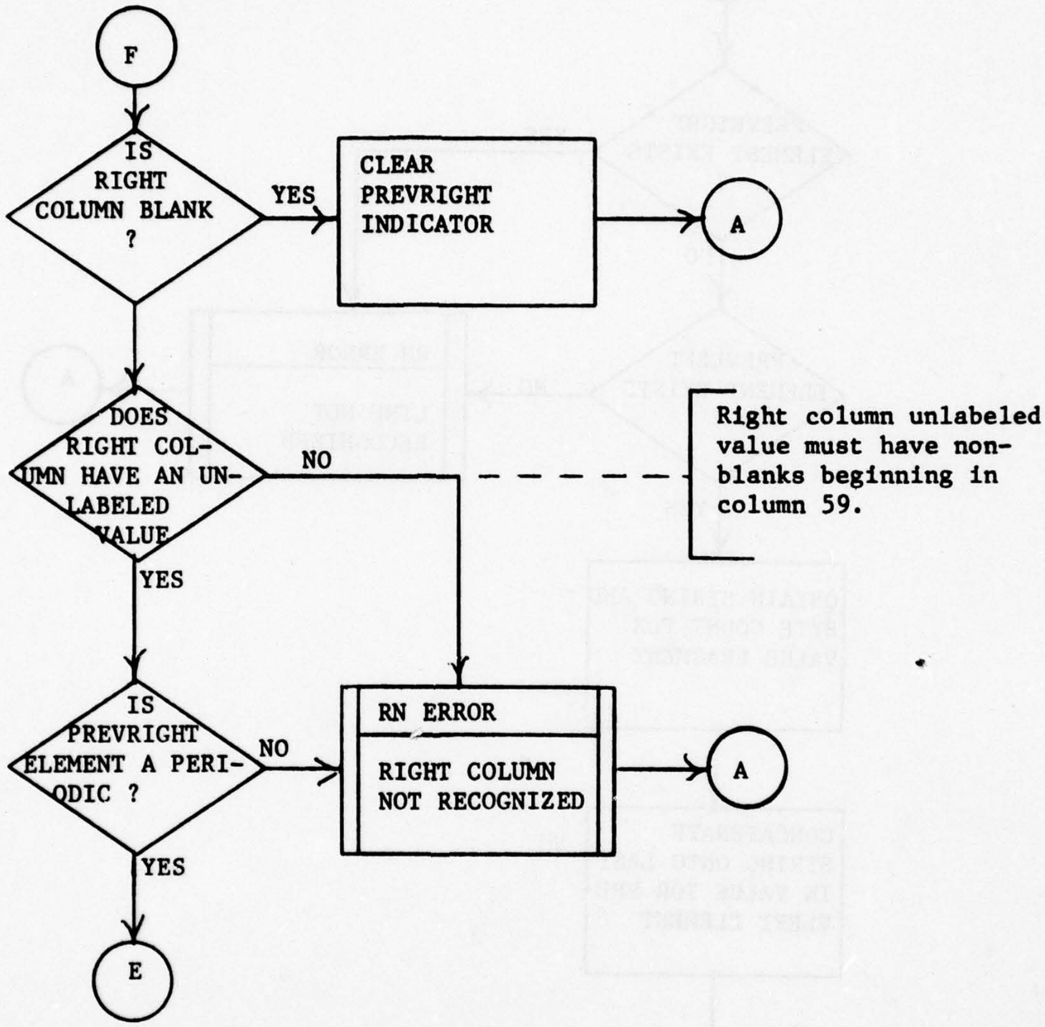


Figure II-16 (continued). Flow Diagram of Line Recognizer.

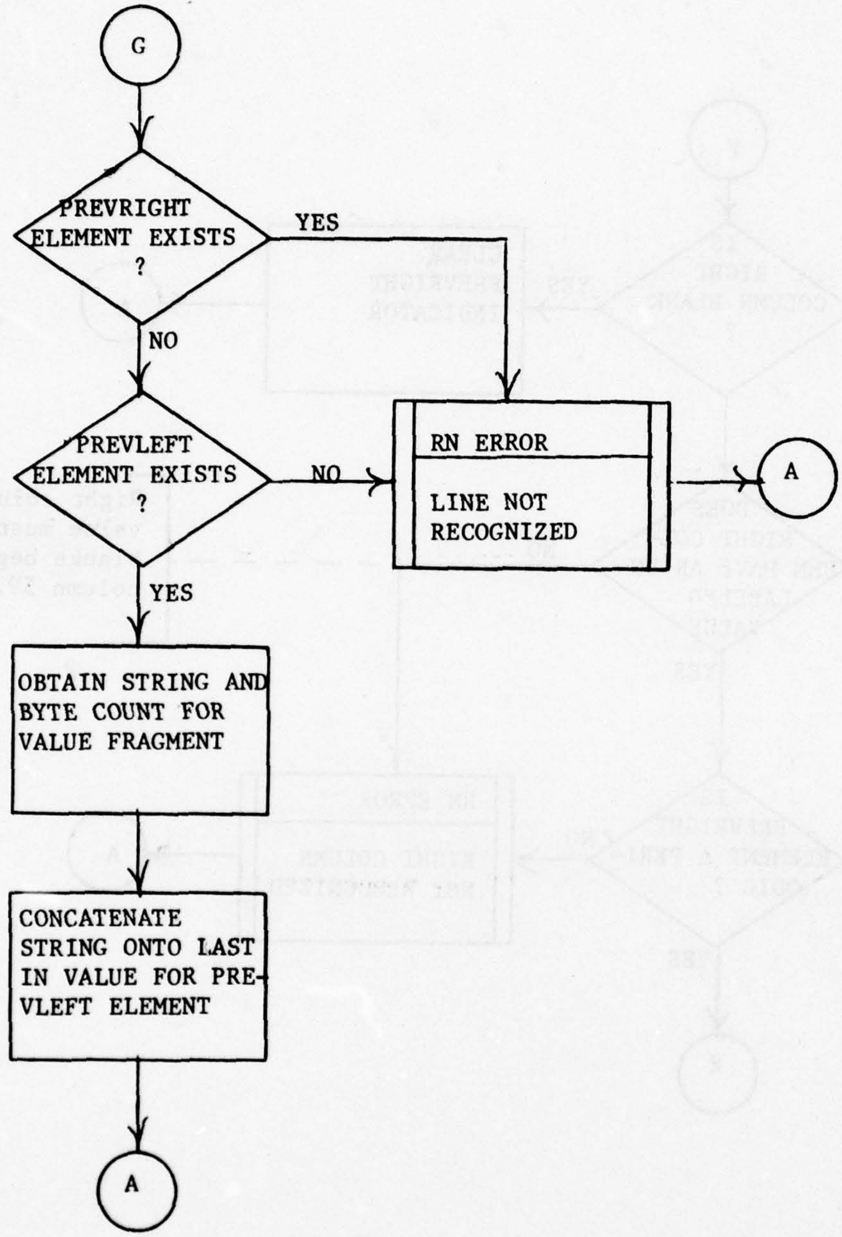


Figure II-16 (continued). Flow Diagram of Line Recognizer.

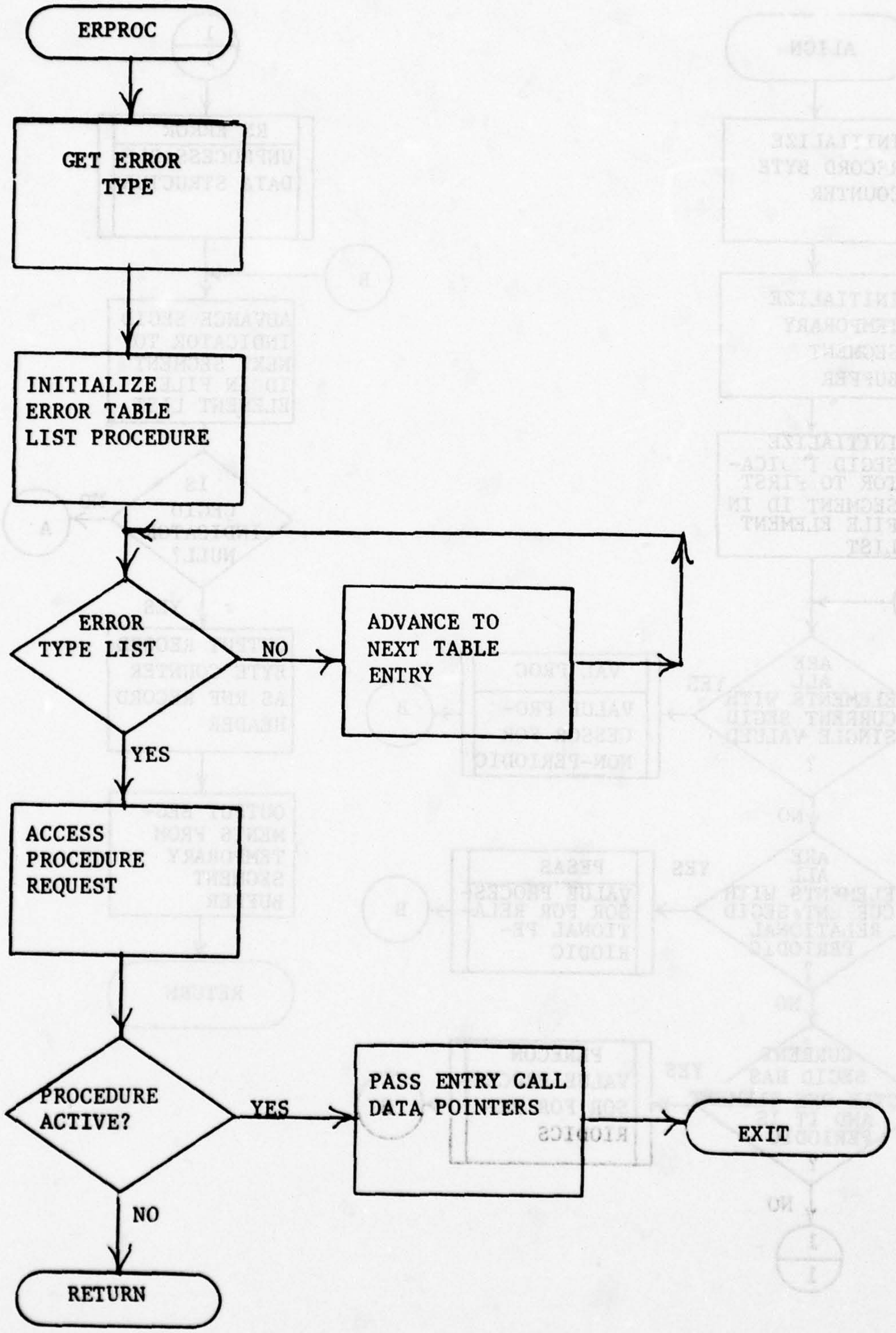


Figure II-17. Flow Diagram for RN Error Processor.

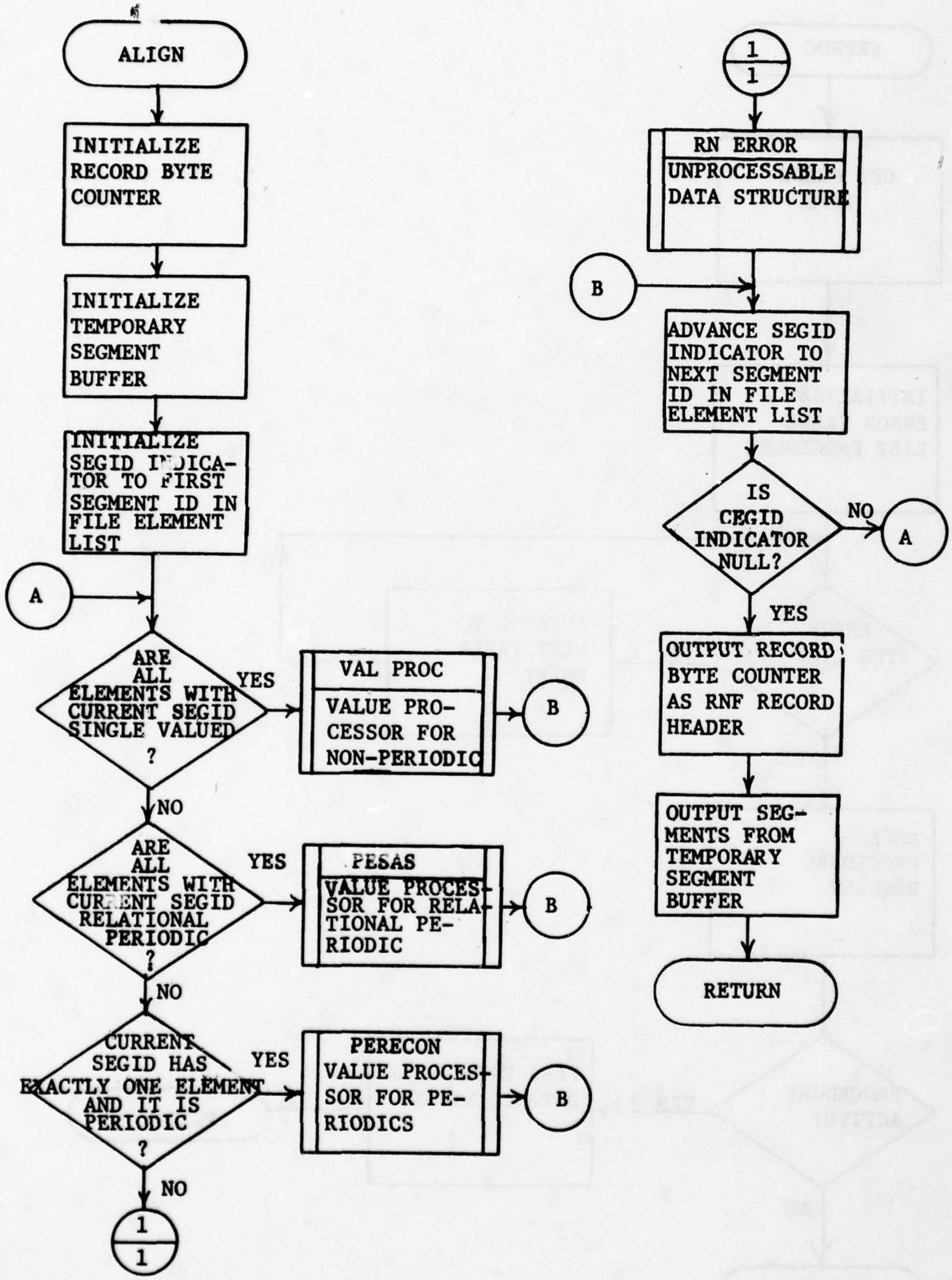


Figure II-18. Flow Diagram for RNF Generator

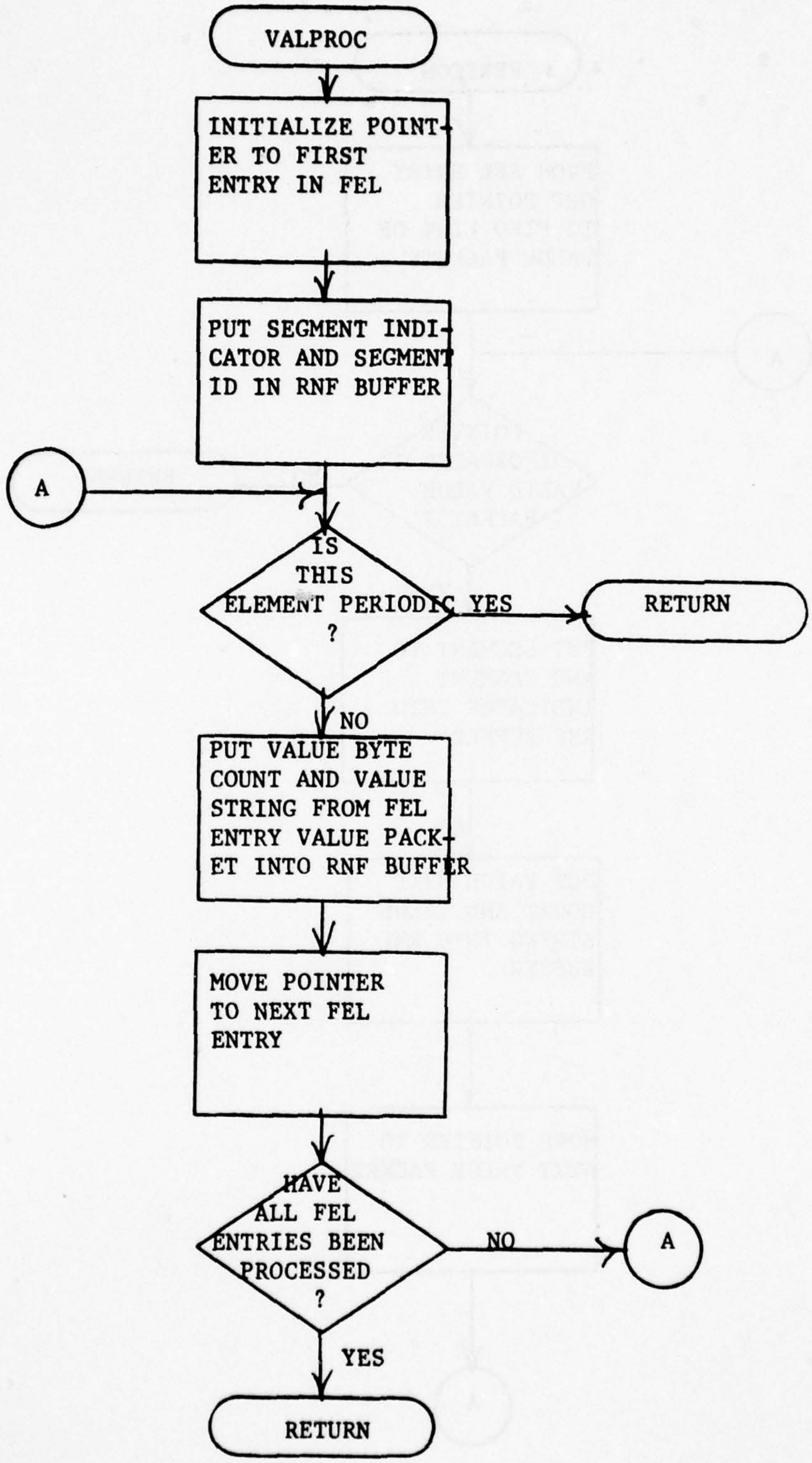


Figure II-19. Flow Diagram of RNF Segment Generator for Non-Periodic Elements

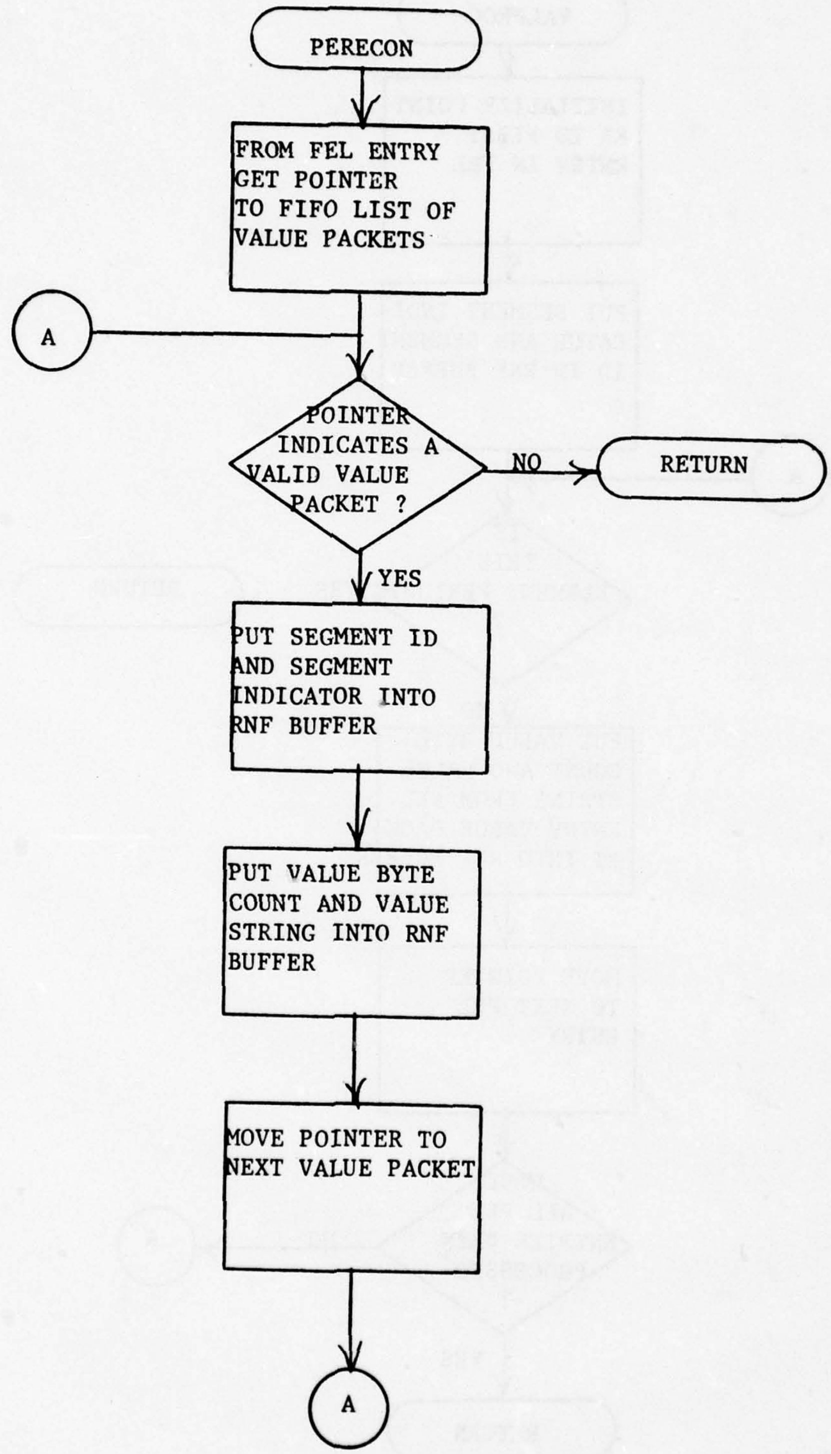


Figure II-20. Flow Diagram of RNF Segment Generator for Periodic Elements

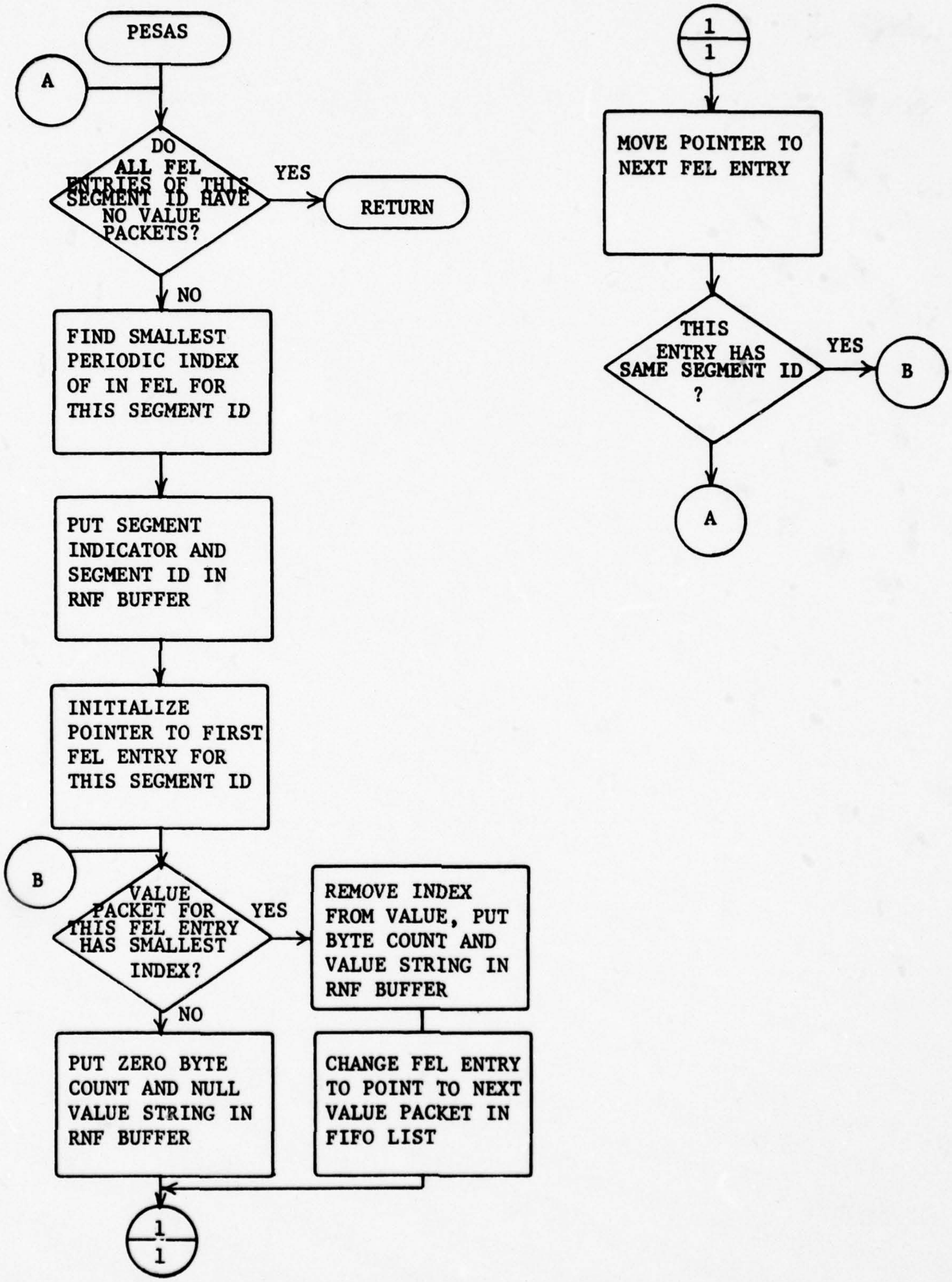


Figure II-21. Flow Diagram of RNF Segment Generator for Relational Periodic Elements

SECTION III

VALUE TRANSLATION INTERFACE

1. VALUE TRANSLATION PROBLEM

Most problems in value translation occur because the same basic information collected at different points has been encoded in different ways. The problem is created when accuracy and information are lost in the data generation and/or data collection process. The data management process serves to propagate copies of data after it has been collected, but obviously the basic data can never be improved, re-collected, or re-generated. The fundamental purpose of any data base is to provide a model of some aspect of the "real world." It is often convenient to perform some data truncation or data compression at the collection point to reduce the aggregate size of the data and thereby reduce the costs of subsequent data management. Such reductions cause problems for several reasons:

- o The reduction is based on assumptions about the eventual analysis and use of the data. Problems arise when other analyses and uses must be considered.
- o The design/choice of a data encodement formula is essentially an arbitrary process with few conventions established or enforced.
- o For convenience in data collection the data encodement scheme is aligned with the data source characterisitcs.

Encodement differences are due to differences in the nature of the source, differences in the purpose of the analysis, and/or the absence of conventions. The differences which occur cover the full spectrum of possibilities. The differences have substantial design implications for a system which seeks to normalize the differences.

a. Value Translation Requirements

The following subsections provide a short survey of examples of differences between value encodement schemes. These differences substantiate the following conclusion, specifically, that the full range of value translation requirements can only be met through the capabilities of a general purpose programming language, and only when it is meaningful to the analyst to perform value translation. The full requirement can be met by a general purpose interface for application programs to read and write files in the format of RNF. This interface is described further in this report in the section on application program interface.

b. Transparent Value Translation

Value translation can be handled transparently in the query and response subsystem of TIIN only in the special case where there exists a one-for-one correspondence between the value encodements, though we offer no mathematical proof of this assertion. This special case can be handled automatically in the TIIN system whenever TIIN detects a difference in value encodements, as in the following cases:

- o Translation of queries from user to network to host frame of reference
- o Translation of response files and error messages from host to network to user frame of reference
- o Translation of response files for input to or output from application programs.

A one-for-one value correspondence may be effected either through a table of "to-from" value pairs or through a pair of algorithms which are inverses of each other.

The preliminary design for a systematic approach to transparent value translation includes the following features:

- o Each item has a data type and a units-of-measurement (UOM)
- o When an item is transmitted to another node, it is converted when there is a difference between the target UOM and the source UOM. It may be converted directly to the target UOM, or it may be converted to a network standard UOM and then to the target UOM.
- o For every non-standard UOM there must be a VTM which converts to a standard UOM and another VTM which converts from the standard UOM to the non-standard.

This set of features seems reasonable for a preliminary design. Subsequent design efforts may incorporate features to handle the practical requirements suggested for the value translation examples. These are:

- Subfields should have all the properties of fields, including UOM
- It should be possible to define a field as a concatenation of existing fields.
- Converter functions must be able to return null values.

2. SHORT SURVEY OF VALUE ENCODEMENT EXAMPLES

A complete survey of differences between currently used value encodements would reveal essentially that virtually all possible differences exist. Such a survey would be beyond the scope of this report. However, a short list of examples is provided to substantiate the above thesis.

a. Confidence Codes

In most cases it is difficult to measure a parameter to one place accuracy when the parameter represents a confidence level, a likelihood indicator, or a probability estimate. Also, it is difficult to analyze or interpret high precision probabilities. Consider "There's a 30% chance of rain" versus "The probability of rain is .3418." Because it is difficult to interpret many levels of probability, many analysts have adopted letter codes limited to as few as three values. Some examples of encodement are as follows:

H : 1.0 - .9	A : 1.0 - .9	A : 1.0 - .8
M : .9 - .1	B : .9 - .75	B : .8 - .6
L : .1 - 0	C : .75- .25	C : .6 - .4
	D : .25- .1	D : .4 - .2
	E : .1 - 0	E : .2 - 0

This practice presents a difficult technical problem: how can a generalized system perform meaningful analysis such as Bayesian estimates on such encoded data?

Equally important is the question of how a user can interrelate such data gathered from distributed data bases, particularly if several different encodement schemes are involved.

b. Range Codes

Ship heights may be coded in one file as A, B, C, D to indicate "less than 20," "20 to 50," "50 to 100," and "over 100," while in another file the actual heights may appear. Such use of range codes results due to excess priority assigned to data storage costs relative to data analysis, and due to limitations placed on data analysis during the data collection process.

c. Country Name Codes

Some data bases use a 2-byte code for the country name (e.g., 'US' for "United States") while other data bases use a 3-byte code of which one byte consists of an area code (e.g., the Middle East countries would all have the same area prefix). The correspondence between the two codes is basically one-to-one, but includes a variety of exceptions. One anomally is that the 3-byte codes include codes which represent areas instead of countries. The area prefix is entered with a null (i.e., "00") country code. This provides a technique for storing aggregate data associated with an area but not associated with a single country.

d. World Area Codes

The globe is divided into "squares" 3°20' on a side. The squares are assigned a code consisting of four non-zero digits, such that all squares with the same first three digits lie in the same 10° latitude longitude square and are numbered clockwise within the 10° square. An algorithm can be used to convert the latitude longitude positions into WAC, and vice versa a WAC can be converted to a range test on latitude and longitude.

WAC = n
corresponds to
$a \leq LAT < b$ AND $c < LONG \leq d$

e. Person Names

Person names are represented with a high degree of inconsistency in large data bases. Some data bases have been purged of explicit person names due to the requirements of the U.S. Privacy Act. It is not possible to construct simple correspondence based on person names.

f. Units of Measurement

When differences of measurement units exist (for example, miles vs. nautical miles, metric vs. English, minutes vs. hundreths) there is a simple conversion between the two encodements, but special attention to round-off problems is necessary since the conversion is not one-to-one. It is useful (but not essential) to avoid conversion to network standard if the user and host encodements are the same.

g. Date Time Codes

One of the few examples of exact one-to-one translation occurs between different encodements for date/time data. For example it is possible to convert into the YYMMDD format from the other formats sometimes used, for example MMDDYY, or DDMMYY, or DDAAAYY, or MM/DD/YY, or YYJJJ using Julian dates, etc. In such cases value translation requires the capability to define the subfields of the target format in terms of the subfields of the formats, a capability which is easily provided using a general purpose programming language.

Time data may involve time-of-day and time-zone considerations, in which case the translation-to-standard requires definition of a subfield in terms of two fields in the source record.

h. Example: Unit ID

The Army Ground Order of Battle files uses a 38-byte Unit ID which consists of a 2-byte country code and a 36-byte JCS Unit ID. This motivates a requirement that subfields have the same characteristics as the field.

3. VALUE TRANSLATION IN QUERIES

The basic algorithm for performing value translation of values which occur in queries has been previously published in the Technical Report for the TIIN Query Intermediate Processor (page IV-2). The basis of the algorithm is that every value which appears in a query is associated with an element (field) name, so the value can be translated simply by accessing the value translation mechanism (i.e., a table or an algorithm) specified in the NAD entry for the element. This algorithm is appropriate if the relational operator in the query is equality or inequality. That is, the algorithm converts a query criteria by changing the element name and the element value from one frame of reference to another:

COUNTRY EQ 'US' becomes NATCODE EQ '201'

This algorithm is rarely appropriate if the relational operator is not equality (EQ) or inequality (NE). With other operators (e.g., LE, LT, GE, GT) the value translation mapping must be order preserving, otherwise the transformed query is not a transparent equivalent. For example:

COUNTRY GT 'US' is not equivalent to NATCODE GT '201'

Examples of order preserving value translations include unit conversions and abbreviations.

a. Basic Constraint for Transparency

The basic constraint to achieve transparent value translation in any query-and-retrieval system is that the value translation be perfectly reversable: in mathematical terms the value translation mapping must be "one-to-one" and "onto." Every value in the user set of value codes must correspond to a unique value in the host set of value codes. The following is a statement of the basic transparency constraint using mathematical notation.

Let Q(X) denote a query on an element X. We say that $X=X_0$ satifies the query Q(X) if $Q(X_0)$ is "true."

Let $\overline{X} = \{X_1, X_2, \ldots, X_n\}$ denote the domain set of discrete values of the element X.

Let $\overline{X}:Q = \{q_1, \ldots, q_m\}$ denote the subset of $\overline{X}$ which satisfies the query Q(X). Read this notation $\overline{X}:Q$ as "$\overline{X}$ such that Q."

With these notations the query can be expressed explicitly as follows:

$$
\begin{aligned}
Q(X) &= (X \varepsilon \overline{X}:Q \\
&= (X=q_1 \text{ OR } X=q_2 \text{ OR } \ldots \text{ OR } X=q_m)
\end{aligned}
$$

Though it may not be practical or efficient in all cases, it is thus possible to re-express a query as entered by the user. This permits a straight-forward conversion of the query under a one-to-one encodement rule.

Let y = E(X) denote a one-to-one encodement.

$$E = \overline{X}:Q = \{E(q_1), E(q_2), \ldots, E(q_m)\}$$

A query R(Y) is considered to be the transparent equivalent of a query Q(X) under an encodement y=E(x) if the following statement is true:

for each x_i in $\overline{X}$ then x_i satisfies Q(x)

if and only if y_i satisfies R(y) when $y_i=E(x_i)$.

For the query Q(x) the transparent equivalent can be expressed as follows:

$$
\begin{aligned}
R(y) &= (y \varepsilon E(\overline{X}:Q)) \\
&= (y=E(q_1) \text{ OR } y=E(q_2) \text{ OR } \ldots\text{OR } y=E\ (qm))
\end{aligned}
$$

For example, if Q(X)=(X GT 'US') then the transparent equivalent of Q(X) can be obtained by first determining which codes in the domain set satisfy Q(X). Assume the following codes satisfy Q(X).

X:Q={'UT', 'XM', 'ZR'}

Then the query Q(X) can be rewritten as

Q(X)=(X EQ 'UT' OR X EQ 'XM' OR X EQ 'ZR')

Now the transparent equivalent can be obtained by replacing the user codes {'UT', 'XM', 'ZR'} by the equivalent host codes {804, 711, 210}. (Note: the codes in this example are fictitious since here the purpose is illustration only.) The transparent equivalent of Q(X) would then be R(y) where

R(Y)=(Y EQ 804 OR Y EQ 711 OR Y EQ 210).

SECTION IV

DISPLAY GENERATOR

1. INTRODUCTION

The function of the Display Generator (DG) module is to convert the response file (RNF) for a query into a user oriented formatted report. This representation can be outputed to the line printer for a hardcopy, to the display terminal, or to a disk data set for use at a later time.

There will be two modules in the DG. The first will be a user interface and the second will be the display generation. The first module will be called the User Interface Module and the second the Response Display Module. In the following two sections the basic design of the two modules will be laid out.

a. User Interface Module

The User Interface Module is simple in design. It will use TTDL for the terminal interface. When the query has been processed the analyst activates the system and the user interface module will call TTDL to display an option menu (Figure IV-1). This option menu will allow the user to either process a file by responding "YES" or to terminate the session by responding "NO".

```
DO YOU WISH TO OBTAIN THE OUTPUT FROM A FILE (YES, NO)?

(A RESPONSE OF "NO" WILL TERMINATE THE SESSION)  YES
```

Figure IV-1. Display to Process Data or to Terminate Session

The analyst is then asked from which files he wishes to view the output. The display will wait until a valid file has been entered. If an invalid file entry has been entered a message will appear informing the analyst of this fact and then prompt him for another file (Figure IV-2).

```
FROM WHICH FILE OR FILES DO YOU WISH TO OBTAIN OUTPUT?
XYZ-7, DIA-5, A730
```

Figure IV-2. Display for Entering Files to be Processed

Should one of the entered files be invalid, a message will appear informing the analyst of that fact, and the remaining files will be processed (Figure IV-3).

```
FILE 'A730' IS AN INVALID FILE ENTRY.

FILE 'XYZ-7', 'DIA-5' WILL BE PROCESSED.
```

Figure IV-3. Display with Valid and Invalid File Entries and Programmer Response

Upon entry of a valid file name, the analyst is prompted for a choice of output format. He must decide at this point if he wishes to use the default values for output or to change one or all of the output choices (Figure IV-4).

The defaults for the output are:

1. Output displayed at terminal
2. No hardcopy output
3. Output displayed in image format
4. Output not saved

```
DO YOU WISH TO USE THE DEFAULT OPTIONS (YES, NO)? NO
```

Figure IV-4. Display of Question for Analyst Decision to Use Default Options

A response of 'YES' will cause the processing of the data according to the default options. Should the analyst wish to change one or all the defaults, an entry of "NO" will cause the questions in Figure IV-5 to appear. The first question will ask if the analyst wants the output displayed at the terminal.

```
DO YOU WANT OUTPUT DISPLAYED AT THIS TERMINAL (YES, NO)? NO

DO YOU WANT HARDCOPY OUTPUT (YES, NO)? YES

DO YOU WISH TO SAVE THE OUTPUT ON A DISK (YES, NO)? YES

WHAT DATA SET NAME DO YOU WISH TO ASSIGN TO THE OUTPUT?
DP: OUTPUT.DAT

WHICH OUTPUT FORMAT DO YOU WISH TO USE? (ENTER "1" FOR IMAGE,
"2" FOR ARRAY): 1
```

Figure IV-5. Displays fcr Default Options

The second question will ask the analyst if he wants hardcopy output. The third question will ask if the analyst wishes to save the information on a disk data set. Should he respond "YES" a display will appear asking for the name of the data set the analyst wishes to receive the output. The final question will ask the analyst whether he wants "image" or "array" formatted output. The difference between these formats is illustrated in Figures IV-7 and IV-8.

Should anything but a 1, 2, or "STOP" appear, Figure IV-6a will appear informing the analyst of an invalid type entry, followed by the question asking the analyst if he wants to see the sample output formats.

```
YOU HAVE ENTERED AN ILLEGAL OUTPUT FORMAT REQUEST
DO YOU WISH TO SEE A DISPLAY OF THE POSSIBLE OUTPUT FORMATS
(YES, NO)? NO
```

Figure IV-6a. Display Informing the Analyst of an Invalid Format Request

An answer 'YES' will cause the output format display to appear (Figure IV-6b), and then the question concerning the choice of output format will appear.

```
THE AVAILABLE FORMATS ARE:
IMAGE:              DATA PRINTED AS FOLLOWS:
    LABEL      UOM    VALUE/DESCRIPTION
      .         .         .
      .         .         .
      .         .         .
ARRAY:              DATA PRINTED AS FOLLOWS:
    LABEL 1             LABEL 2            LABEL 3          . . .
    UOM 1               UOM 2              UOM 3
    ----------          ----------         ----------      ----------
    Value 11            Value 21           Value 31         . . .
      .
      .
      .
    Value 1M            Value 2M           Value 3M         . . .
```

Figure IV-6b. Output Format Display

This cycle of questions will be reported again, and if a valid entry is not entered, the default output format will be used, and a message will appear informing the analyst of this fact (Figure IV-6c).

```
THE DEFAULT FORMAT "IMAGE" WILL BE USED.
```

Figure IV-6c. Display Informing the Analyst of Invalid Format Request Entries and Use of IMAGE Format for Output Display

An escape mechanism is provided in case the analyst wishes to terminate the session. Should the need arise, a response of "STOP" to any one of the questions will terminate the program.

b. Display Generator Module

The DG module receives the commands from the interface module and processes the files according to the choices indicated.

The DG module will scan the files which are to be queried and will extract the value names, unit of measurements and value description from each file.

After each file has been completed, the DG module will display the output in one of two formats. The first of these is referred to as an IMAGE format. This will direct the DG to print the information out according to record (Figure IV-7). The second format is referred to as an ARRAY format. This information groups variable descriptions that are common to various records and then prints out the data in arrays, with the variable names and units of measurement as column headings (Figure IV-8).

When the DG module has reached an EOF, a message will appear at the end of each file output and the DG module will return to the Interface Module.

The flow chart for the Interface and DG module are included and described on the following pages.

OUTPUT FROM FILE: AIRPORT RUNWAYS DATE: 18 OCTOBER 1977

LABEL	(UNIT OF MEASUREMENT)	VALUE
AIRPORT	(NAME)	SEQUOIA
LENGTH	(FEET)	3Ø5Ø
ANGLE	(DEGREES)	8Ø
AIRPORT	(NAME)	HAYWARD
LENGTH	(FEET)	34ØØ
ANGLE	(DEGREES)	85
AIRPORT	(NAME)	HAYWARD
LENGTH	(FEET)	528Ø
ANGLE	(DEGREES)	1ØØ
AIRPORT	(NAME)	MODERTO
LENGTH	(FEET)	5ØØØ
ANGLE	(DEGREES)	95
AIRPORT	(NAME)	MEADOWS
LENGTH	(FEET)	57ØØ
ANGLE	(DEGREES)	12Ø
AIRPORT	(NAME)	MEADOWS
LENGTH	(FEET)	38ØØ
ANGLE	(DEGREES)	8Ø

END OF RESPONSE FILE: AIRPORT RUNWAYS

Figure IV-7. Example of "Image" Style Formatted Output.

OUTPUT FROM FILE: AIRPORT RUNWAYS DATE: 18 OCTOBER 1977

AIRPORT (NAME)	LENGTH (FEET)	ANGLE (DEGREES)
SEQUOIA	3Ø5Ø	8Ø
HAYWARD	34ØØ	85
HAYWARD	528Ø	1ØØ
MODERTO	5ØØØ	95
MEADOWS	57ØØ	12Ø
MEADOWS	38ØØ	8Ø

END OF RESPONSE FILE: AIRPORT RUNWAYS

Figure IV-8. Example of "Array" Style Formatted Output.

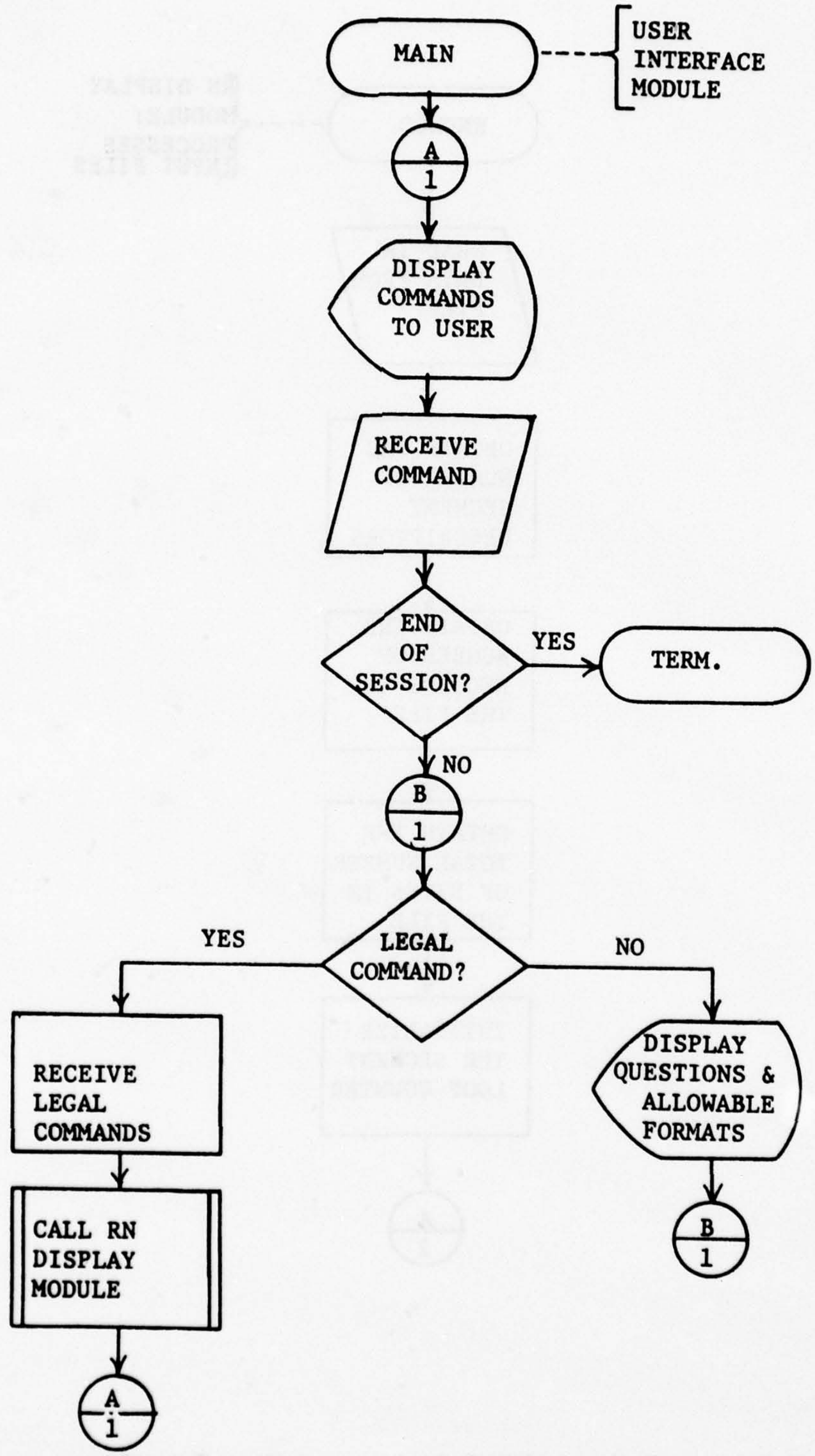


Figure IV-9. MAIN of User Interface Module

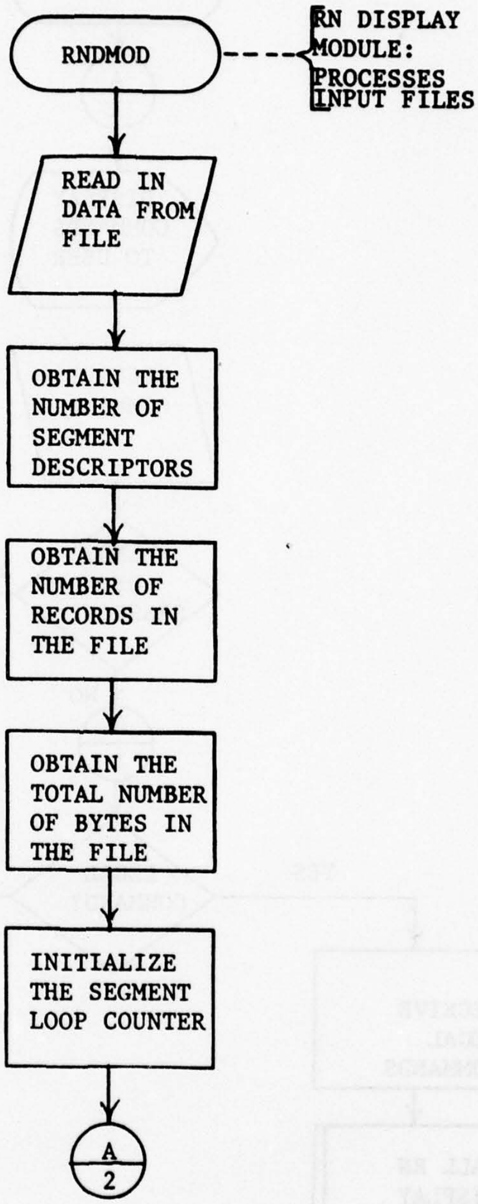


Figure IV-10. RNDMOD Routine of User Interface Module (Page 1 of 8)

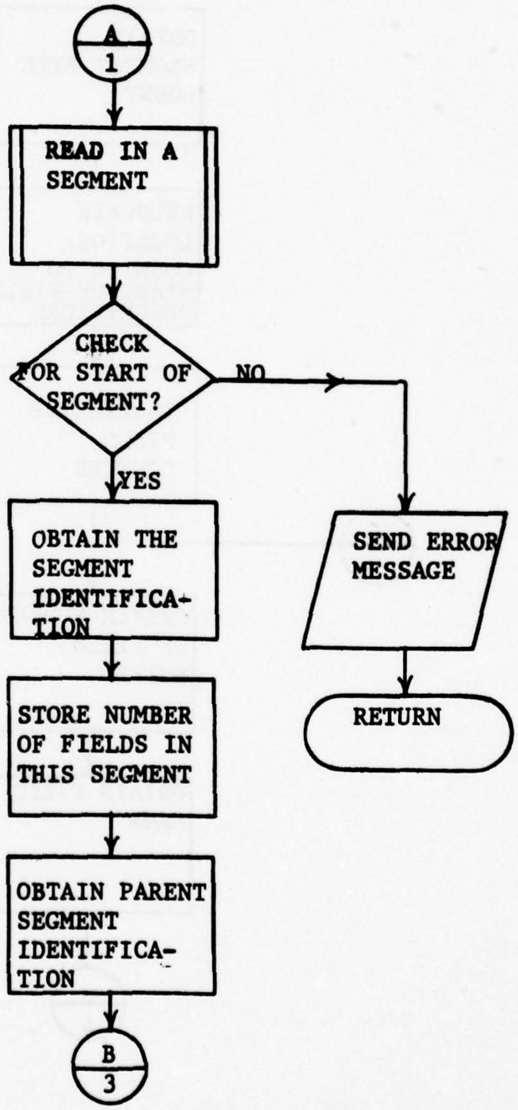


Figure IV-10. RNDMOD Routine (Page 2 of 8)

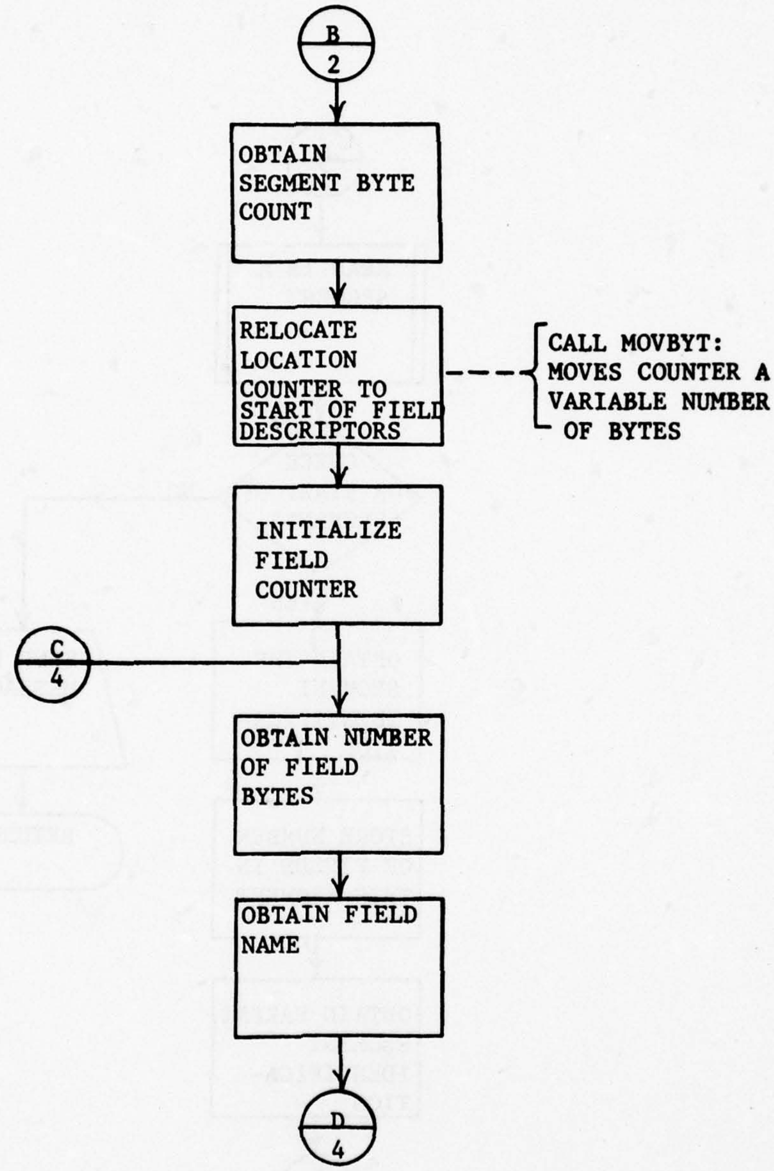


Figure IV-10. RNDMOD Routine (Page 3 of 8)

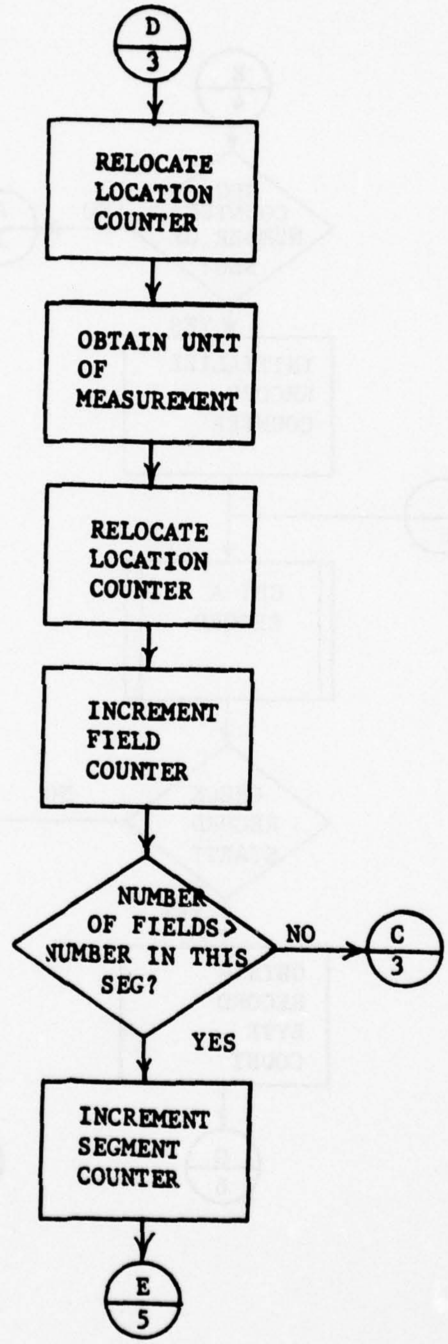


Figure IV-10. RNDMOD Routine (Page 4 of 8)

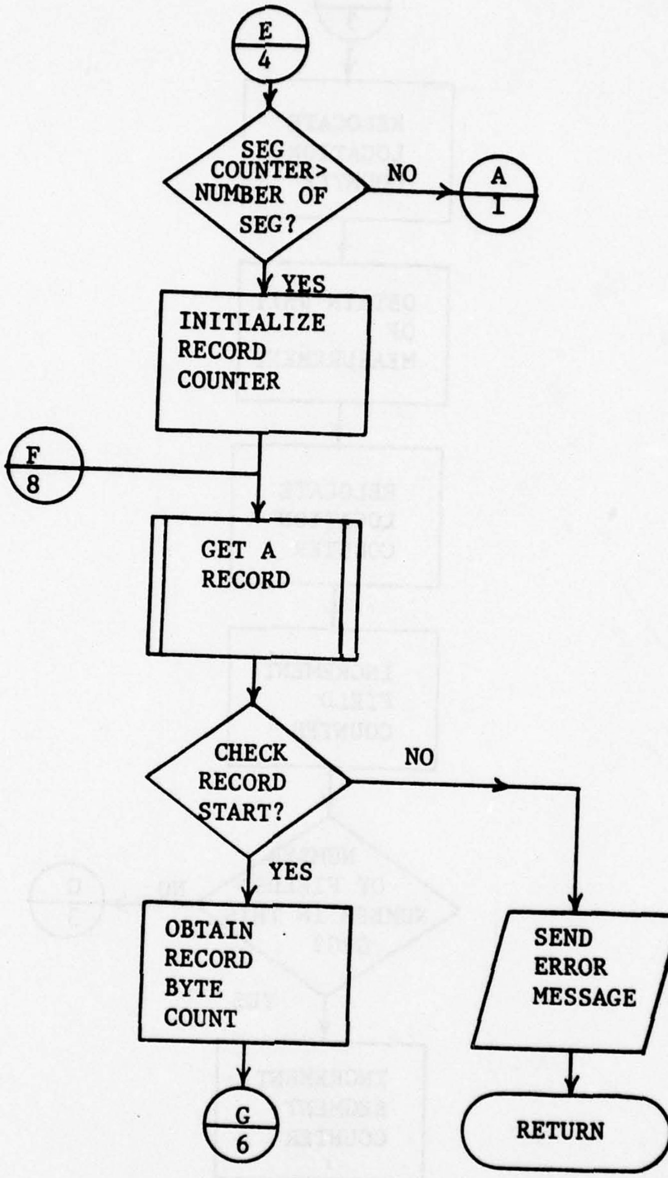


Figure IV-10. RNDMOD Routine (Page 5 of 8)

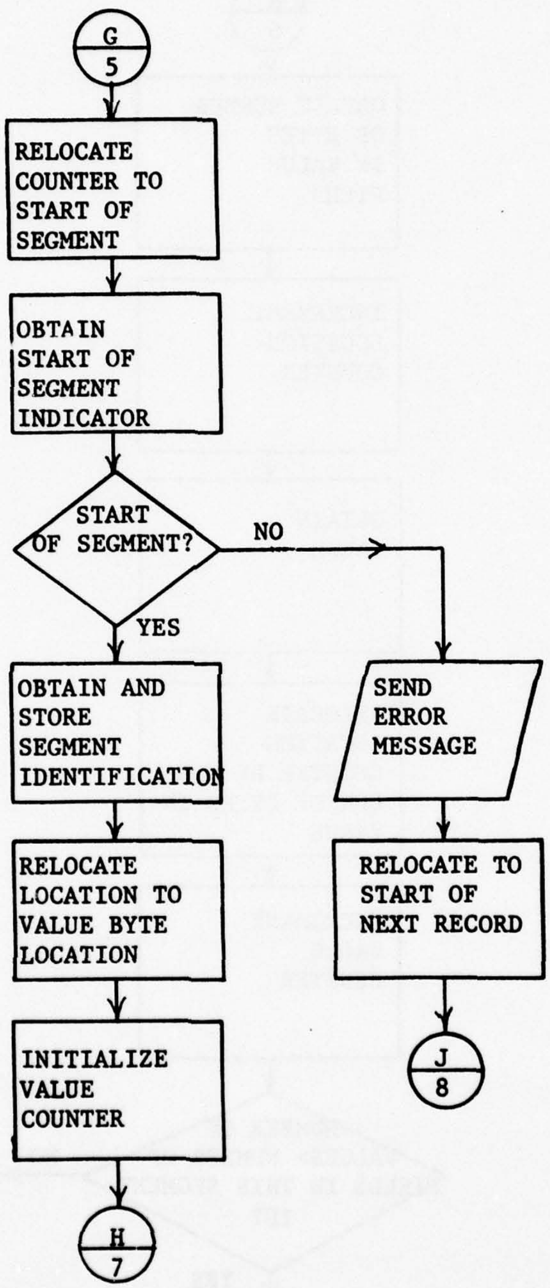


Figure IV-10. RNDMOD Routine (Page 6 of 8)

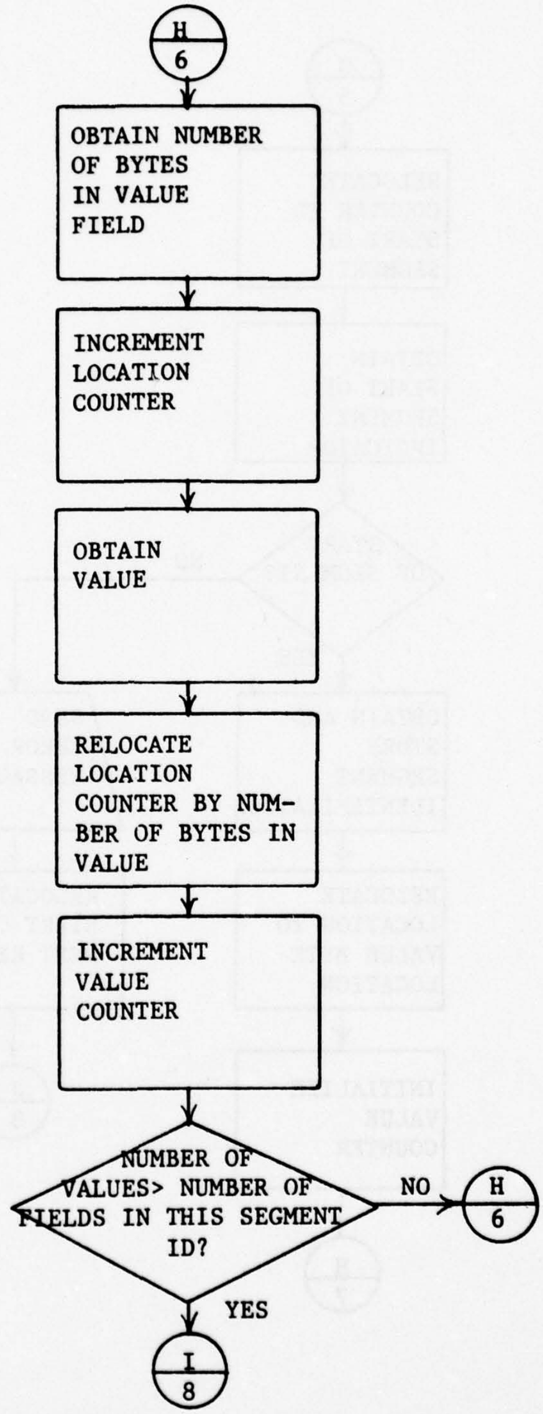


Figure IV-10. RNDMOD Routine (Page 7 of 8)

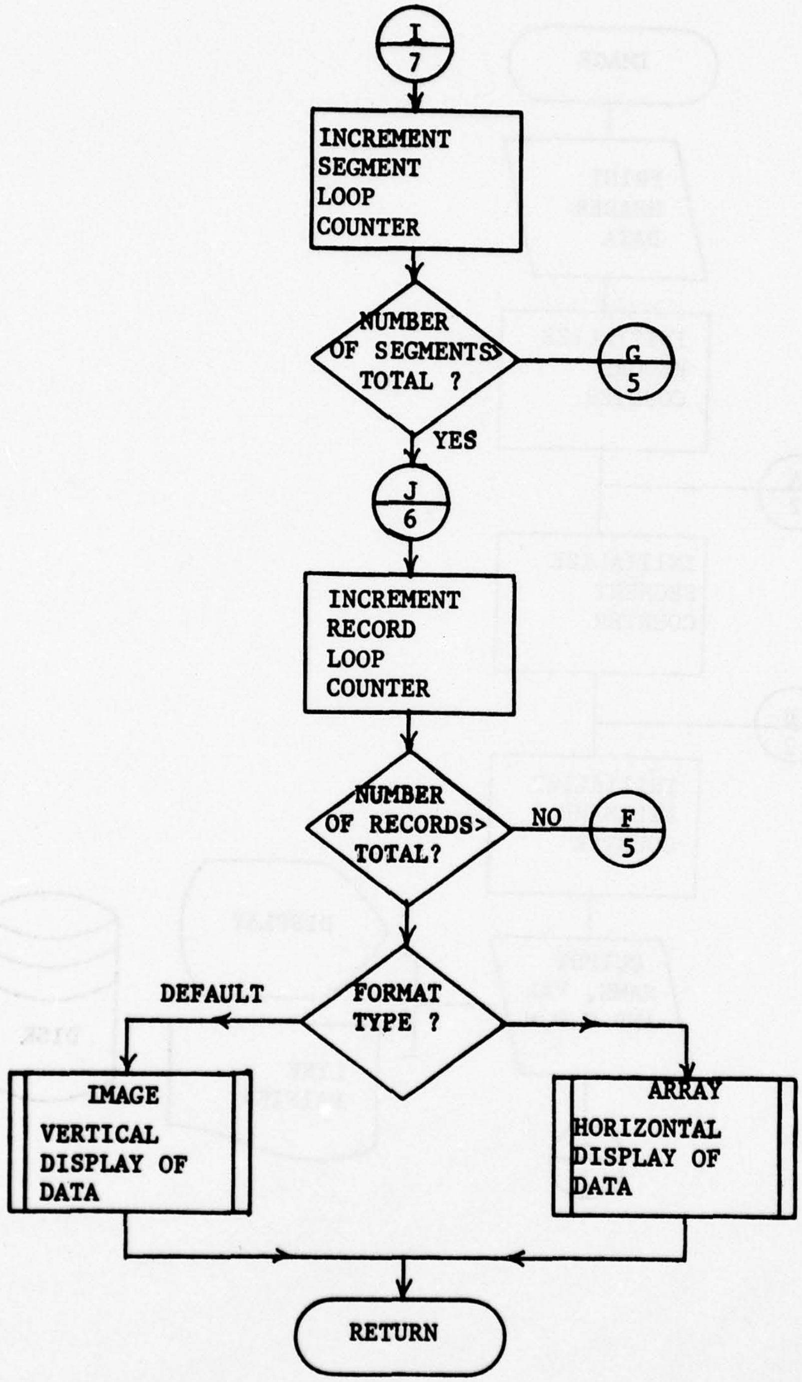


Figure IV-10. RNDMOD Routine. (Page 8 of 8)

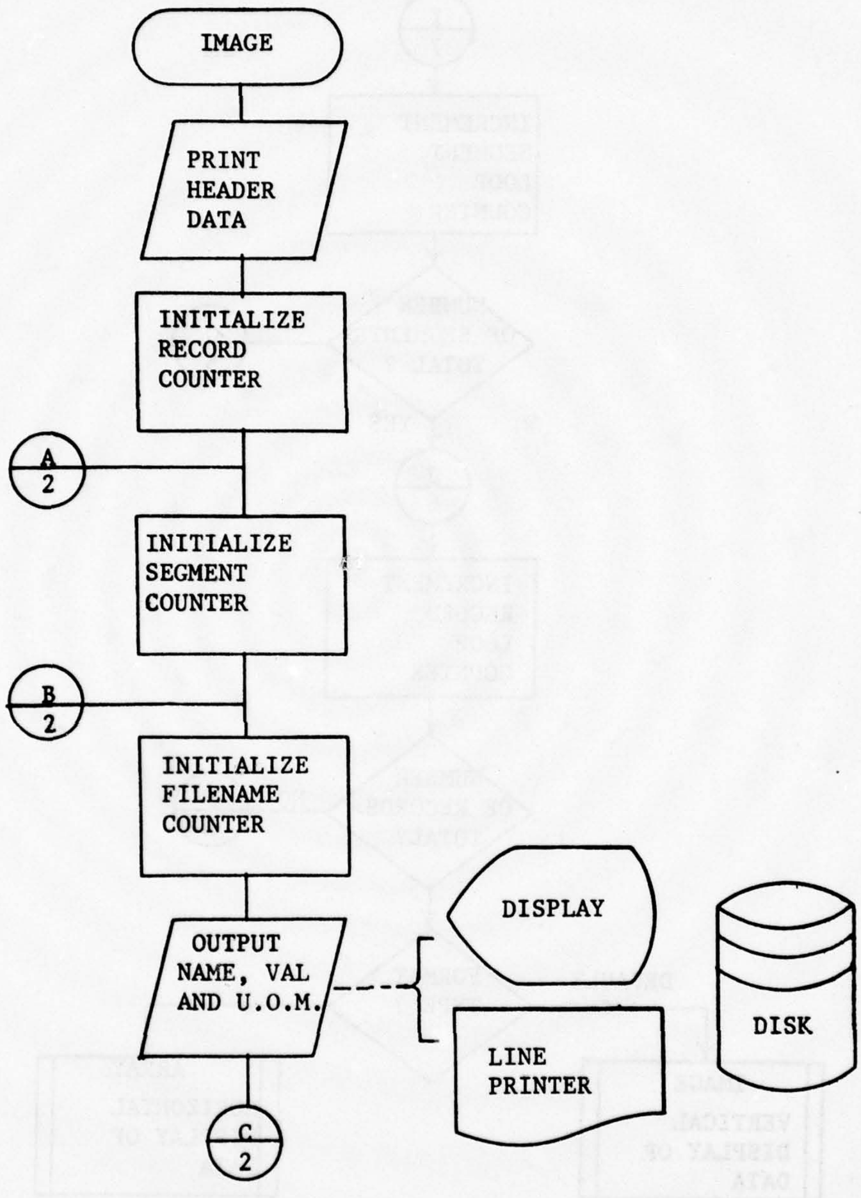


Figure IV-11. IMAGE ROUTINE (Page 1 of 2)

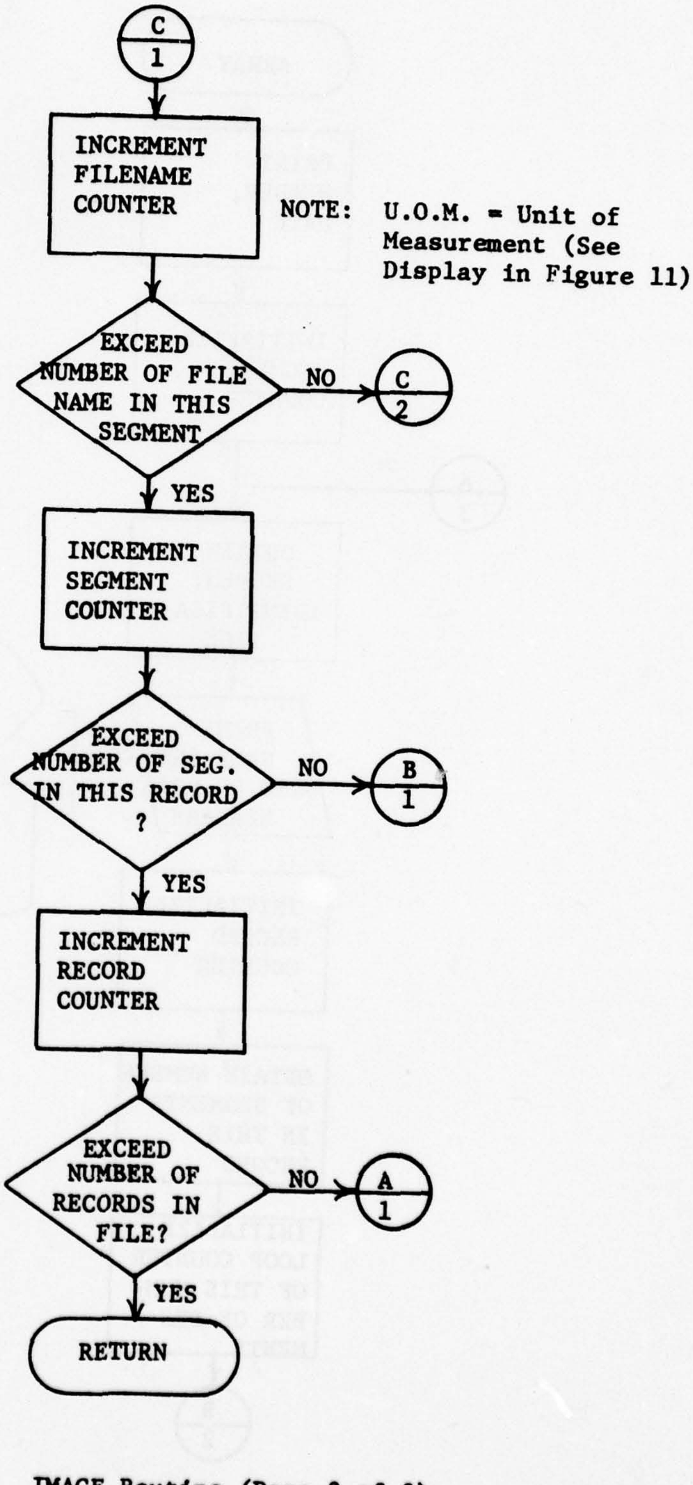


Figure IV-11. IMAGE Routine (Page 2 of 2)

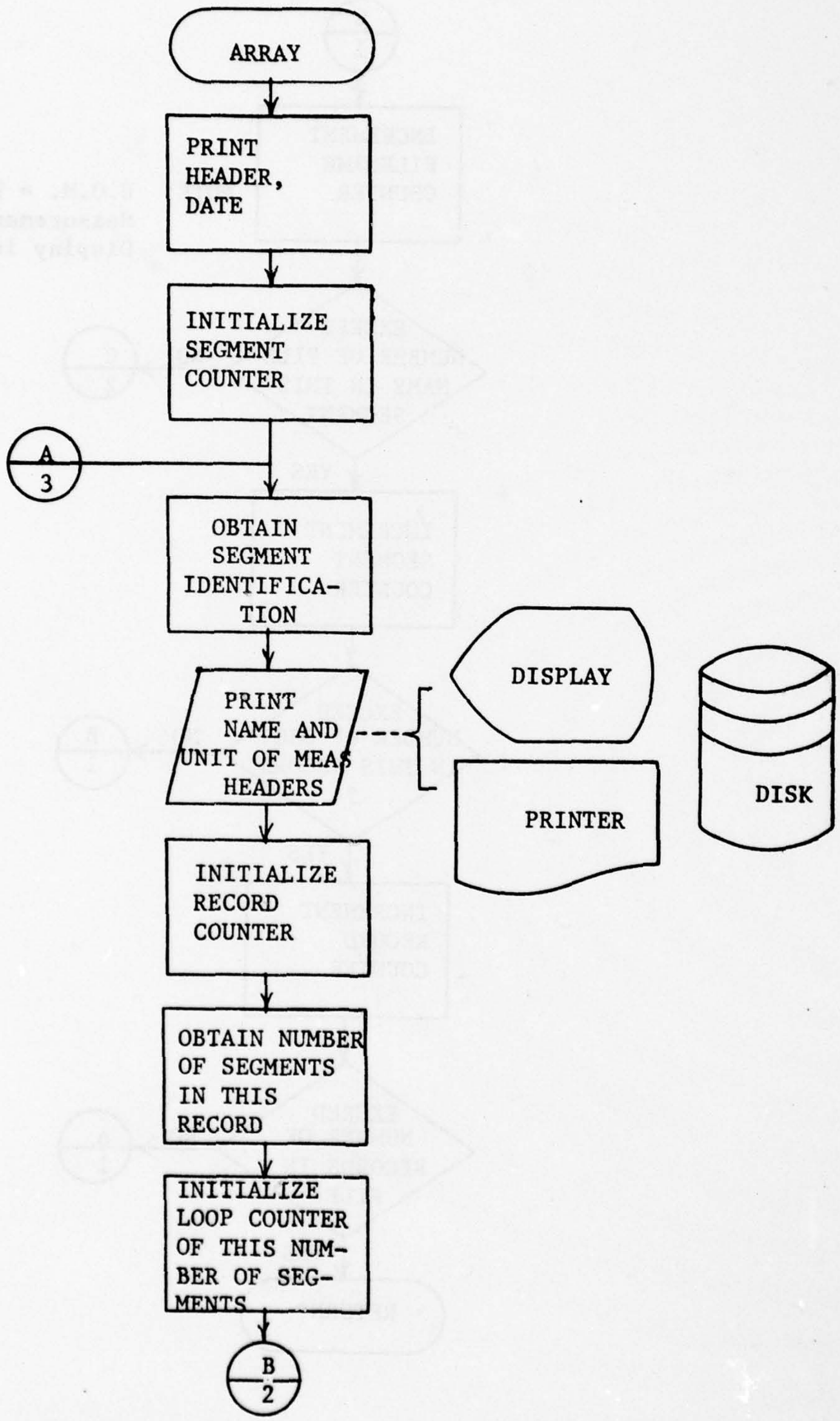
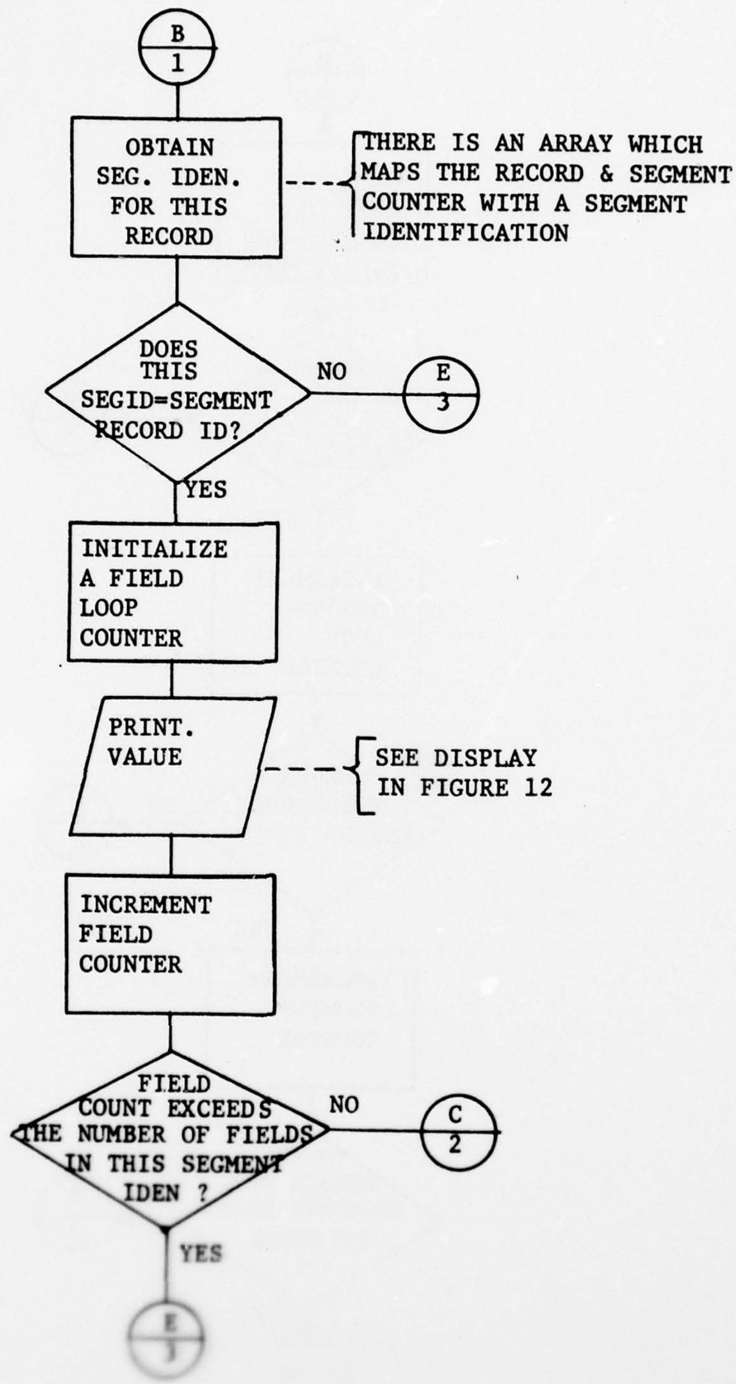


Figure IV-12. ARRAY Routine (Page 1 of 3)


B
1
OBTAIN SEG. IDEN. FOR THIS RECORD
THERE IS AN ARRAY WHICH MAPS THE RECORD & SEGMENT COUNTER WITH A SEGMENT IDENTIFICATION
DOES THIS SEGID=SEGMENT RECORD ID?
NO
E
3
YES
INITIALIZE A FIELD LOOP COUNTER
PRINT. VALUE
SEE DISPLAY IN FIGURE 12
INCREMENT FIELD COUNTER
FIELD COUNT EXCEEDS THE NUMBER OF FIELDS IN THIS SEGMENT IDEN ?
NO
C
2
YES
E
3

[illegible]

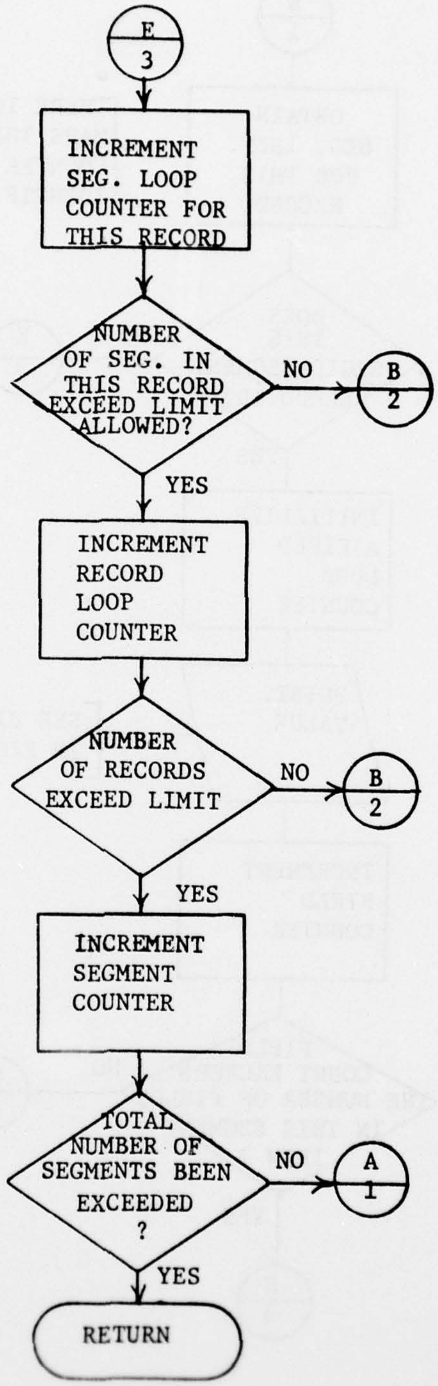


Figure IV-12. ARRAY Routine (Page 3 of 3)

SECTION V

APPLICATION PROGRAM INTERFACE

1. INTRODUCTION

The provision of value translation mechanisms in the TIIN system takes two forms:

- o Mechanisms which can be handled transparently in queries and in response files, and thus can be specified by the DBA and performed by the TIIN system without the explicit knowledge or control of the user.
- o Mechanisms which cannot be handled transparently and thus must be explicitly invoked by the user.

The interface for non-transparent VTM is the primary subject of this section. The interface for transparent VTM is described in the section on value translation; but a brief description is included below in this section.

The TIIN requirement for a flexible interface for user specification of value translation mechanisms is essentially a requirement for an interface to permit application programs to access TIIN response files. Several different interfaces are possible:

- o The lowest level interface deals with the discrete fields which comprise the RNF. This interface is the most efficient and is intended for the TIIN utility routines.
- o Another interface level for applications programs deals with the data as it exists in the response file, organized into segments of variable length data fields.
- o For many types of programs, especially those written in languages such as FORTRAN or COBOL, the best interface is one in which the data is organized into fixed format segments with fixed length fields.

The primary subject of this section will be an interface for application programs to access response files consisting of variable length fields.

The interface for fixed format files is not discussed in detail in this report. However, it remains an important area of possible future development of TIIN, an area which may be postponed for the present. Future selection of an affiliated DBMS will be the vehicle for achieving fixed format access to TIIN response files.

a. Interface for Transparent VTM

The interface for transparent VTM is described in the section on value translation. Briefly, a transparent VTM is either a table of pairs of values or a pair of algorithms, and its function is to convert values from one UOM (unit of measurement) to another UOM and vice versa, such that if value A in one UOM corresponds to value B in the other UOM then the VTM maps A to B in one direction and B to A in the other direction.

The functional role of transparent VTM in the TIIN system is to perform translation of values when queries are forwarded to the host and when response files are received from the host. The objective is that the user sees only the field names and value codes of the user frame of reference, while the host sees only the field names and value codes of the host frame of reference.

The interface for Transparent VTM is based on the UOM attribute of each field. The VTM is triggered whenever there is a change in UOM accompanying a change in field names. Changes in field names occur when queries are converted from user to network or from network to host, or when response files are converted from host to network or network to user. When a change in field name occurs, the associated value code (if any) must be changed so that both the field name and the associated value code are always from the same frame of reference.

b. Interfaces for Non-Transparent VTM

Any value code conversion which is not one field to one field and one-to-one value mapping is too complex to be handled in a transparent manner in the TIIN system. There might be a few exceptions to this observation (e.g., concatenations). However, the essential conclusion is that the TIIN requirement for a flexible interface for user specification of VTM can be satisfied primarily through an interface for user-written application programs to access (read/write/update) TIIN response files.

The viewpoint in this report is that any process which inputs a response file and outputs a response file can be considered to be a value translation mechanism (VTM). The user/programmer must be thoroughly familiar with the host value codes and the host field names to formulate the query and to program the value conversion

of the response file from the host frame of reference to the user frame of reference.

The basic requirements to pass a data file to a program are as follows:

- o A complete description of what format the program expects: for each field, the name, size, and relative offset.
- o A complete description of the data file format: for each field, the name, units, and relative offset.
- o Correspondence between the response file and the desired input file for the program: for each field the corresponding fields.

(1) Interface for TIIN Utility Programs

For maximum efficiency it is necessary to have a file interface which provides access to the response file essentially "as is," that is, with no conversion between what the file contains. The essential restriction on the program is that it must be able to process data items of unpredetermined size, units, and codes, where the item descriptor is contained in the response file together with the actual items.

(2) Interface for Variable Length Values

Since the response file is organized as a self-descriptive hierarchical file of variable length segments comprised of variable length values, it is useful to have an interface to permit application programs to process the data as it is organized in the response file.

This interface is the primary subject of the latter part of this section.

(3) Interface for Fixed Format Files

For COBOL programs and other programs which require fixed length records with fixed length fields, an effective interface can be achieved via a general purpose utility program which converts a TIIN response into a fixed format file. The converter utility uses an external description of the application data format expressed in a data description language (DDL).

For COBOL programs the data description can be extracted from the program source code (specifically, the data division). This provides an external description which can be used to specify the target format for a TIIN response file, provided that the fields in the two formats correspond in a straightforward manner.

If the response file and the application file have different hierarchical structure and/or different sort keys it will be necessary to restructure and/or sort the application file. A sort is often necessary when a value translation occurs on the key item.

The interface for fixed format files is not discussed further in this section. However, it remains an important area of possible future TIIN development, an area which has been postponed for the present in anticipation of selection of a TIIN affiliated DBMS.

(4) Interface for Dynamic Data Declaration

It is possible to construct application programs so that the object modules are not affected by changes in external file structure. What is necessary is that the program explicitly declare all its assumptions about each file element. In general there are two points at which the declaration might occur: at load time or at execution time. There is a complete parallel between file declaration and element declaration. When a file is declared (i.e., "opened") at execution time the applications program specifies the file name and the assumed characteristics such as record size, sequential vs. direct access organization, variable vs. fixed length, and buffer size. A virtual file number is used to coordinate all subsequent references to the file in the program. By analogy a field would be declared by associating a virtual element number with the assumed characteristics of the element specifically:

- o Element name
- o Number of bytes
- o Type or picture
- o Units of measurement
- o Segment name.

It may appear to be inefficient to declare each element separately, and to access each element value via separate

subroutine calls. However, the application program need access only the items actually required, which would allow the TIIN software to avoid converting the whole of each record. The primary design consideration is generality, specifically, to use the same support software for applications programs as for TIIN modules such as the report generator, the collation routine, and the value translation routines.

This interface will not be discussed further in this section. It is mentioned here as one of the possibilities which was considered but not pursued.

2. UTILITY INTERFACE

The function of the utility interface is to perform all operations concerned with details of storing RNF contents in buffers.

The format (RNF) for response files is specified in Figure II-1. The RNF is accessed using the basic file open, read, and write primitives of the operating system. The contents of the RNF are transferred to the program on a field-by-field basis, sequentially, in the same order as the fields appear in the response file. An internal pointer is used to indicate the position of the next byte to be accessed in the RNF buffer. This internal pointer is automatically updated in the operation of the RNF buffer management routines. The following variables are used consistently in the calling sequences:

LUN - The logical unit number corresponding to the file and referenced in all file operations from OPEN to CLOSE.

LEN - The number of bytes in the field.

STR - Byte array which contains the bytes of the field being accessed.

EX - Exit error code (0 = no error, 1 = data error, -1 = end-of-file).

The following routines constitute the TIIN utility interface for read and write operations for the RNF buffer.

INPTR (LUN) - Initialize the buffer read/write pointer to access the first byte in the RNF header.

GETSTR (LUN,LEN,STR,EX) - Move LEN bytes from buffer to string array STR.

GETVLN (LUN,LEN,STR,EX) - Move a variable length string to string array STR, put length in LEN.

SKPSTR (LUN,LEN,EX) - Skip over LEN bytes in buffer.

SKPVLN (LUN,EX) - Skip over contents of one variable length field in buffer.

PUTSTR (LUN,LEN,STR,EX) - Move LEN bytes from string array STR to buffer.

PUTVLN (LUN,LEN,STR,EX) - Move LEN bytes from string array STR to buffer and store as a variable length field.

a. Disk Layout for Response File

A response file is arranged on blocks of 512 bytes each. The blocks are linked so that a sequential pass through a response file can only be accomplished by obtaining the virtual block number (VBN) of the "next block" from the link field of each "current block." This linking permits the TIIN system to store multiple response files in one logical file and it permits the logical ordering of the records to be different from the physical ordering.

The layout for a block consists of a header, a data area, and an unused portion, as illustrated in Figure V-1. The header includes the following information:

- VBN of next block.
- Number of bytes of data on the block
- Subfile identifier
- Layout version number
- Pointer to beginning of first record (if any) on the block.

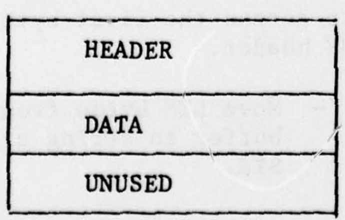


Figure V-1. Block Layout for RNF

3. BASIC PRIMITIVES TO ACCESS RESPONSE FILES

The basic primitives to provide access to response files are as follows:

Open File

Select Record

Get Segment Identifier

Read Segment into String Array

Delete Segment

Insert Segment

Replace Segment.

The principal consideration in the design of these primitives is that the application programs need not know the structure of the response file in advance. Without any prior knowledge of the structure the program can open the file and obtain the item names and types and the segment composition and interrelationships from the record descriptor tables formed by the open file primitive. The record can be processed segment by segment by using the primitive for reading the identifier of the next segment, followed by the read segment primitive to actually select and read the next segment.

a. Open File

The primitive to open a response file performs the following actions:

- o Accepts a logical unit number from the calling program
- o Reads the file header to obtain the Query ID, the File ID, the User Node ID, the number of segments, and the number of levels
- o Reads the segment descriptors and field descriptors for the file.

The record count and byte count are also retrieved from the header, and are used internally in the utility interface to verify that all the data has been processed. Figure V-2 illustrates the descriptor tables formed by the open file primitive. Possible errors in the open file are as follows:

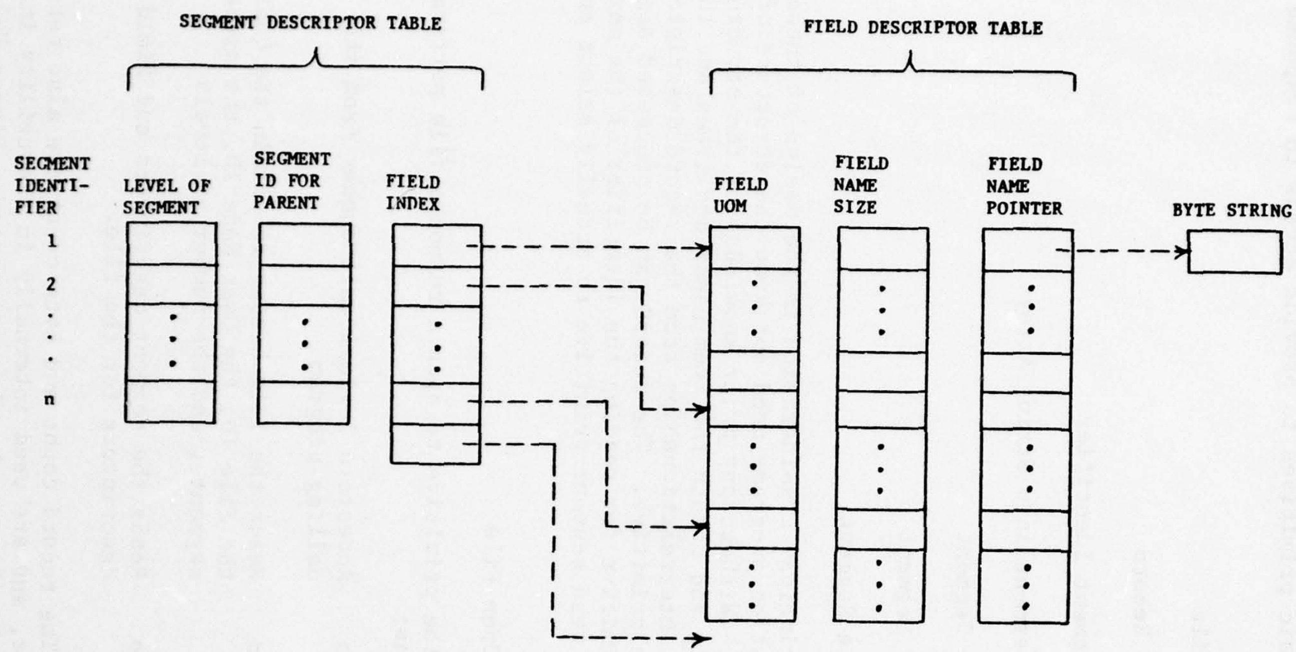


Figure V-2. Descriptor Tables for Segments and Fields (Items) of a TIIN Response File.

- o File not available
- o Inadequate space for descriptors
- o Invalid LUN.

b. Select Record

The primitive to select a record performs the following actions:

- o Clears the parameters for the previous selected record
- o Reads the record header, including the record byte count.

The record byte count is used when the next record select command is executed to access the next record.

The errors possible in the select record primitive are as follows:

- o Record not found
- o Invalid LUN.

c. Get Segment Identifier

The primitive to read the identifier of the next segment performs only that function. The next buffer block will be accessed if necessary to access the next segment.

The errors possible in the get segment identifier primitive are as follows:

- o No more segments in current record
- o Invalid LUN.

The primitive to get the identifier of the next segment is intended to be used in conjunction with the primitive (below) to find and read segment into a string array.

d. Read Segment into String Array

The primitive to read a segment performs the following functions:

- o Searchs and selects the next segment with segment identifier in the record
- o Builds an array of pointers to strings with one pointer for each field of the segment.

The errors possible in the read segment primitive are as follows:

- o No more segments with specified identifier
- o Invalid LUN.

e. Delete Segment

The primitive to delete a segment from a record performs the following actions:

- o The current selected segment with the indicated identifier is removed from the current selected record. All dependent segments are also removed.
- o The current selected record is rearranged on its physical blocks to represent the record which results after the deleted segment is removed.

If the indicated segment is the root segment then the entire record will be deleted.

The errors possible in the delete segment primitive are as follows:

- o Invalid LUN
- o Indicated segment not currently selected.

f. Insert Segment

The primitive to insert a segment performs the following action:

- o The indicated segment to be inserted is added to the current selected record in a position before the current selected segment (if any) with the same identifier.
- o The inserted segment becomes the currently selected segment.

The errors possible in the insert segment primitive are as follows:

- o Invalid LUN
- o Segment to be inserted does not have currently selected parent
- o Segment to be inserted must be adjacent to sibling segments with the same identifier.

g. Replace Segment

The primitive to replace a segment in a record performs the following action.

- o The current selected segment with the indicated identifier is replaced (overwritten) by the indicated segment. The replacing segment inherits the parents, siblings, and dependents of the replaced segment.
- o The current selected record is rearranged on its physical blocks to represent the record which results after the segment has been replaced.

The errors possible in the replace segment primitive are as follows:

- o Invalid LUN
- o Indicated segment not currently selected.

APPENDIX A

DEVELOPMENT BACKGROUND

1. TOSS BACKGROUND

Over the past four years, Terminal Oriented Support System (TOSS) concepts and implementation methodologies have undergone research and development by INCO, INC., of McLean, Virginia, for the Rome Air Development Center (RADC). The objective is to provide intelligence analysts with transparent access to computer files, remote data bases and other analysts world-wide through a computer-based communications network. TOSS development has proceeded along lines which seek to extend the concepts of transparency and distributed processing to all relevant aspects of the intelligence ADP environment.

The approach taken by INCO in implementing these concepts has been to build upon existing technology, such as that used in the NMIC Modernization project. Briefly, TOSS is a multifaceted information management and handling system which aids in integrating distributed intelligence resources in a cohesive, world-wide network. It is a hardware/software system employing minicomputers as stand-alone, terminal-oriented processors connected to local host computers to form nodes, which are in turn interconnected by communication links to form a widespread network. The principal components of TOSS include TOSS Exchange Center (TEC) providing network communications control for bulk data and conversational message traffic; Terminal Independent Support System (TISS) providing the capability for on-line analyst interaction through transparency of communication protocols; TOSS Information Management System (TIMS), a data management system providing on-line capabilities for network file access and development and query of hierarchical files; and finally, the Terminal Transparent Display Language (TTDL) facilitating the development of applications programs and stressing the concepts of terminal transparency.

2. TIIN DEVELOPMENT

The TIIN development effort seeks to extend the transparency and distributed processing concepts of TOSS to include query language transparency and data base transparency. TIIN will be based on the hardware/software environment of TOSS with its access to a widespread network of data base management systems with different query language, different technologies for file structure, different conventions for encoding data.

The eventual aim of the TIIN is to allow the user/analyst to converse with the network without regard for file structure and host dependent considerations. The user should feel as though he is accessing information available to him at this local node when, in fact, he is accessing remote data bases, perhaps world-wide.

AD-A054 308 INCO INC MCLEAN VA F/G 9/2

TRANSPARENT INTEGRATED INTELLIGENCE NETWORK - RESPONSE NORMALIZ--ETC(U)

APR 78 P STYGAR F30602-77-C-0030

UNCLASSIFIED INCO/1088-1277-TR-55-D(F) RADC-TR-78-74 NL

2 OF 2
ADA
054308

END
DATE
FILMED
7 -78
DDC

The TIIN development follows an integrated, step-by-step approach consisting of successive development stages. Each of the stages of TIIN development provides a discrete advance in user capabilities. Advances in system capability require corresponding technological advances. The development of a technology base for TIIN is planned to be systematic and cumulative; most technological features are applicable to all TIIN development stages and are enhanced at each level.

The user capabilities added at each TIIN development stage are described below:

a. Transparent Access to a Local Data Base

The analyst may obtain a description of entry to and use of a local minicomputer data base without specific knowledge of the data base management system.

b. Transparent Access to Distributed Homogeneous DBMS

Local and remote data bases having the same data management system appear as a single data base from the viewpoint of analyst accessibility. Capabilities such as stored queries and distributed standing queries are to be integrated into the network.

c. Transparent Access to Distributed Heterogeneous DBMS

Data base resources available to the analyst are extended to include data bases accessed by data management systems other than a single (common) DBMS, but which are still within the minicomputer network.

d. Integration of Host Computer and Foreign Networks

The network available to the analyst is extended to include data bases in host systems and foreign networks which are connected to the minicomputer network via its nodes. The concept of a network user is extended to include analysts not connected to one of the host systems.

e. Integration of Work Stations Having Intelligent Terminals

Network capabilities are fully integrated with specially designed features and capabilities afforded by an intelligent terminal implemented as an analyst work station in the intelligence community.

3. GUIDING CONCEPTS

The development of the Transparent Integrated Intelligence Network has proceeded in accord with several major principles. They include transparency, data base integration, data sharing, and distributed processing. Each of these ideas will be dealt with in this subsection.

a. Transparency

This concept implies that detailed system knowledge is not required by the user to access and operate network elements. In a transparent environment, analysts can access information from unfamiliar data bases using procedures with which they are familiar. Query translation, data translation, and reconciliation are provided by special software modules, relieving analysts and network elements from requirements for mutual familiarity.

The notion of transparency, as applied to an information processing system, refers to those system characteristics which allow a user to ignore, indeed be unaware of, complexities and diversities which are not relevant to his purpose. For example, problem oriented languages such as COBOL and FORTRAN make machines language transparent to their users.

The concept of transparency as used in the TIIN effort is best communicated in terms of levels of transparency.

(1) No Transparency

Without transparency the analyst must use separate terminals independently connected to each external system. He must use the precise system access procedures as well as that query language associated with the data base he is accessing (the native language of the data base).

(2) Communications Transparency

At this level, the analyst would be able to retrieve data from external systems at his own analyst station, that is, communications and line protocols would be made transparent to the analyst, as well as logon procedures to the external systems. The analyst would still be required to use the data base access procedures (query languages) of the various external systems in order to retrieve data.

(3) Query Language Transparency

At this level, the analyst is able to retrieve data from diverse data bases and data files using a single query procedure from his own terminal. He does not have to know the query language on the particular external data base. There are, however, demands made on the analyst's time and knowledge inasmuch as he must be aware of the dispersion, structure and conventions for data bases being accessed.

(4) Data Base Transparency

At this level, dispersion of data resources among the various files and separate data bases is transparent to the analyst. Data is retrieved using one data access procedure from the analyst's terminal; computer software is used to separate the single query into separate queries to be directed at various data bases and files containing the requested information. In addition, the disparity among similar data element names and coded data field values is resolved.

b. Integrated Data Base

The intelligence community shares data requirements. The accuracy and completeness of an intelligence analysis presupposes that all relevant data be available to the analyst.

The concept of an integrated DBMS is concerned with providing all users with access to all data bases on a network. The concept presupposes the existence of a common data access language which is implemented system-wide. Integration may be achieved in many different ways: merge all data bases, provide a common DBMS with distributed data bases, or provide query language transparency and data base transparency for existing systems.

c. Delegated Production

The concept of delegated production seeks to minimize data maintenance costs by designating specific sites with non-overlapping responsibilities for data collection. A natural corrolary of designating specialized data production centers is that analysts must be able to access and cross correlate the data from the non-overlapping data bases. The volume of data updates and the size of the data bases involved make it non-cost-effective to maintain copies of the data bases at the data analysis sites separated from the data collection sites.

d. Distributed Processing

The concept of distributed processing seeks to allocate computer resources on a network basis so that response time and overall network utilization are optimized. Balanced availability of computer resources permits the community's computational requirements to be satisfied even when some processors become temporarily unavailable to the network.

4. THE ANALYST

There are two distinct categories into which user analysts

may be classified. The first comprises those who have a non-computer background (perhaps in area studies, languages, the liberal arts) who use a technician to interact with the system. The other class of user analysts greater knowledge of ADP and can be expected to exploit the system to a greater degree than the first group. This latter group includes data specialists, file sponsors, and data base administrators.

Though both groups may benefit from the development of transparency oriented processing, it is probably the non-computer oriented analysts who may benefit most from increased user-orientation of data access. To enhance productivity and to utilize the analyst's problem-oriented training, the analyst cannot be unduly concerned with the mechanics of data retrieval or with the arbitrary differences between query language, files structures, and data codes on a distributed network of diverse intelligence resources.

a. Analyst's Task

The intelligence analyst is responsible for assessing information related to critical and complex situations which have far-ranging political and military significance both nationally and internationally. The analyst frequently must work under considerable time pressure. To support his analysis and judgemental capabilities, a large volume of diverse information exists in multiple and uncoordinated sources. Analysts have traditionally developed highly personalized and often very sophisticated manual techniques for accessing information based on experience and knowledge of sources. While the analyst may resort to rather sophisticated "shoe-box" techniques for storage of retrieved data, the main concern is not data storage or retrieval, but analysis.

The analyst's basic mission is to monitor and interpret intelligence information and report it in a timely and effective manner for further assessment at decision-making levels. In fulfilling this mission, he performs the following activities.

- Monitoring messages, indicators, and events
- Establishing and maintaining data files
- Seeking information from other analysts and data files
- Preparing finished intelligence reports
- Joint assessment of current intelligence situations
- Verifying and enhancing information through cross-correlation.

These activities are conducted within the context of three functional levels related to the urgency of the situation at hand. For the indications and warning (I&W) analyst these are:

- Watch Function - monitoring indicators and assessing current events
- Current Intelligence - actively tracking significant events and crisis situations
- In-depth Analysis - developing background information and analyzing information with long-term implications.

These functions are keyed to the operation of command centers and the formulation of precise military decisions based on the best possible intelligence information. Within this context, the analyst's objectives are timely and accurate assessment and subsequent reporting of complete intelligence data. To accomplish these basic, but difficult, objectives the analyst has very real requirements which can be met through data processing support.

b. Data Analysis Requirements

There is a strong and continuing need for automated support to provide the analyst with ready access to data. A large measure of automatic data processing support is being developed using facilities already in place. However, these facilities are geographically dispersed and reside on independent hardware and software systems.

A major need is on-line access to data bases which are not part of the analyst's immediate system. To be useful to the analyst, ADP support must make minimal demand on him to learn access techniques. The ideal ADP support should unburden the analyst from an already substantial data processing workload. This requires the implementation of interactive, integrated networking concepts with sufficient translating and routing software to permit the analyst to access external systems in his own familiar terms. The provision of simplified, real-time access to remote data bases would greatly aid the analyst in cross-correlation, data assessment and reporting functions.

The volume and variety of intelligence data, and the ways in which it may be stored and handled in the intelligence community are great. Merely to monitor the vast array of information provided by sensors and other data collection devices within the analyst's area of specialization can be an immense job; the cross-correlation of data is staggering. Requirements for automatic data processing support clearly exist both with regard to the acquisition of data <u>and</u> its later manipulation for analytic purposes.

Data processing needs are critical to the nature of the indications and warning analyst's mission, functions, and activities. Automation can support not only the monitoring activities in which the analyst is frequently engaged, but will greatly assist the analyst's performance in crisis situations.

Analyst data needs amenable to the application of ADP technology generally focus on the access and manipulation of data. Both access and manipulation have several facets. Access requirements are broad and specific: the analyst has a great interest in a wide variety of sources and indicators, as wll as specific information requirements. Since the analyst desires to keep abreast of developing situations, single elements of intelligence are sometimes critical; timeliness, validity, and reliablility are also very important. The data manipulation needs of the analyst focus on the requirement to rapidly process and format large volumes of diverse information. By way of illustration, some access and manipulation needs are: immediate updating of key indicators, corroboration of hypotheses with initial intelligence multi-source cross-correlation support, program generation for formatting unfamiliar data packages, real-time requests for specific information, and queries to experts outside of the analyst's field of specialization.

APPENDIX B

DESCRIPTION OF THE TIIN SYSTEM

1. INTRODUCTION

The purpose of this appendix is to provide a concise description of the function of the components of the Transparent Integrated Intelligence Network (TIIN). For development purposes the components are grouped into five subsystems to be specified and developed in the following order:

TIIN/TILF	data description and directory management
TIIN/QIP	host selection and host query generation
TIIN/RN	report normalization and report formatting
TIIN/TAP	analyst aids processor and query normalization
TIIN/QDNC	query distribution and network interface.

These subsystems and their components are shown in Figure B-1. This figure will be referenced throughout this appendix. The figure illustrates the distinction between user-dependent components and host-dependent components. It is assumed that host dependent components exist only at the host node, and user-dependent components exist only at the user node, since this minimizes the number of copies of the host dependent components in the network. *One alternative is to place all components at* the user node, which would reduce bottlenecks at the host node as a tradeoff for duplicating all host-dependent components at every user node.

Briefly, the key to understanding the TIIN system is the concept of normalization, which refers to the use of internal standards to serve in the absence of standards between the interconnected host computers. Three areas of normalization can be considered to be the province of the TIIN system specifically, query normalization, report normalization, and data structure normalization. Other areas of normalization, such as host access protocols, packet format, and terminal interface, are the province of the network software/hardware system. It is assumed that TIIN will operate on the PDP-11/45 under RSX-11D, using the software components of the Standard Software Base (SSB) and the Terminal Transparent Display Language (TTDL). The network interface will be based on WICS Common Format (WCF) or its successor.

Normalization allows the user to have many of the advantages of standardization without causing the hosts to entail the disadvantages of standardization. Through normalization the enormous range of differences between the host systems becomes transparent to the user.

a. Data Structure Normalization

The TIIN internal convention for data structure presupposes that the host data base consists of a collection of interrelated hierarchical files. A record in a hierarchical file consists of a collection of hierarchical dependent segment, each segment consisting of a fixed number of variable length single valued fields. Each segment except the root segment has a single parent segment, and each segment type may occur a variable number of times in its dependency relation with its parent segment. There is one root segment per record, and all segments in the record are either directly or transitively dependent on the root segment.

The normalized data structure is stored in the Network Access Directory (NAD) by the Data Base Administrator (DBA). The NAD contains all the data descriptions needed by the TIIN components. The hierarchical relation between segments is represented by indicating the identifier of the parent segment for each segment.

b. Query Normalization

The TIIN internal convention for queries, called Query Normal Format (QNF), is essentially reverse Polish notation with all names and values explicitly represented and flagged to indicate whether they are in the user, network, or host frame of reference. At the user node the QNF is called Data Request Intermediate Format (QTF) to denote the internal representation used for performing algebraic transformations to convert the query into the host query language. The QTF and the associated transformations are substantially different for each host.

c. Report Normalization

The TIIN internal convention for reports, called Response Normal Format (RNF), consists of a file descriptor followed by a sequence of records. The response file is completely self-descriptive and is thus completely independent of the files described in the NAD. The RNF version of the report is suitable for further processing: it may be formatted as a user report, or input to an applications program, or stored as a file in the analyst data base.

2. TERMINAL ACCESS PROCEDURES (TAP)

Components of the TAP subsystem appear in the upper middle and left section of Figure B-1. The subsystem includes the Analyst Aids Processor, and the Query Language Processor.

The TAP subsystem is designed to provide a convenient and natural user interface method. The functions of TAP are 1) to describe network data resources to a user/analyst, 2) to support the user in the construction of legal queries, and 3) to allow for rapid, interactive dialogue with the network.

a. Query Language Processor

The Query Language Processor will provide the analyst the capability of expressing a query via a simple formal language. The module will check the syntactic correctness of the query and will produce the normalized version (DRIF) of the query for the Query Intermediate Processor (QIP) at the user node.

b. Analyst Aids Processor

The descriptive information which is maintained in the Network Access Directory for use by TIIN components is also useful to the analyst in identifying the precise data requirements. The Analyst Aids Processor displays the contents of the NAD in a user-oriented format.

NAD information is displayed by subject category with prose descriptions of files related to subjects the analyst selects, prose descriptions of data elements belonging to selected files, and properties and attributes of selected elements.

3. TRANSPARENT INTELLIGENCE LANGUAGE FACILITY (TILF)

The components of the TILF subsystem appear in the upper part of Figure B-1. This subsystem is devoted to describing the contents of host data bases and keeping these descriptions up-to-date for real-time access from any node in the network. The subsystem includes the User Data Description Processor, the Host Data Description Processor, and the Network Access Directory.

a. Network Access Directory (NAD)

The Network Access Directory (NAD) is the repository for all information required by the TIIN system about the contents of network data bases. Individual versions of the NAD will be created and maintained at each network node, and either all or part of each local NAD must be able to be easily transmitted to other network nodes on request. This feature will enable each node to periodically gather and consolidate information from its own and other NADs, consolidate the information, and use it to locate data throughout the network.

The largest segment of a local NAD will be a file, each entry of which contains descriptive information about a local data element, a network standard element (which may or may not have a corresponding local element), or a local-usage name for a network data element. Other locally-maintained NAD files will contain descriptive subject and routing information for the parent files and parent data bases of local elements and legal values or ranges of values for local elements.

Data Element entries must be accessible via two modes of user access. Normally, specific data elements will be "found" when an analyst expresses an interest in a particular subject; however, from frequent use of the network system, the analyst may choose to refer by name to elements, thus bypassing perusal of subject categories. The information about a data element includes:

- o Normalized name
- o Element type (string or numeric)
- o Length and format information
- o Identifier of parent file
- o Identifier of parent data base
- o Value translation algorithm or table.

b. User Data Description Processor

The User Data Description Processor is one of the User modules in the TIIN. Unlike the Query Language Processor and the Analyst Aids Processor, this processor is password protected. It is intended for use only by the User Data Base Administrator (DBA), not by TIIN analysts.

The User Data Description Processor creates and updates the User NAD. Its function is to provide a means for the User DBA to define that portion of the network data which is available to the local analysts.

c. Host Data Description Processor

The Host Data Description Processor is one of the Host TIIN modules. The Host Data Description Processor is privileged and password protected. It is intended to be used by a Host DBA only.

The Host Data Description Processor creates and updates the Host NAD. Its function is to provide a means for the Host DBA to describe to the network those files and those elements which are potentially accessible to all User nodes. In addition, it provides a means for the Host DBA to perform updates whenever necessary and to display information contained in the NAD.

Both the User Data Description Processor and the Host Data Description Processor run in the interactive mode with their respective DBA. Neither DBA is required to know the exact structure of the NAD. Rather, what is important is that they understand the structure of the data bases which they are defining.

4. QUERY INTERMEDIATE PROCESSOR (QIP)

The QIP subsystem appears in the center of Figure B-1. This subsystem deals with host selection, element name translation, and query translation. The user-dependent portion of QIP (User QIP) is called the Translation and Data Base Selection module. The host dependent portion of QIP (Host QIP) is called the Host Query Language Generator.

The functions of the User QIP are 1) selection of host data bases and files containing the elements requested by the user, as indicated in the DRIF, and 2) translation of element names and values from the user to the normal TIIN frame of reference.

The host QIP translates the user query from the TIIN normal format (QNF) into the host DBMS query language.

The sequence of actions through which the query will be processed includes:

- Identification of those files which contain the requested data elements
- Substitution of network-standard element names for synonymous user element names
- Creation of equivalent versions of a query for addressing different files which may contain similar data
- Creation of sequences of queries when several files must be searched in sequence to retrieve data, or when several different search verbs are necessary to process all the search criteria

- o Reconciliation of distinctive functions in the QNF not provided for in the host language
- o Translation of network standard element names into element names in the host frame-of-reference
- o Generation of the host version of the user query.

5. QUERY DISTRIBUTION EXECUTIVE/NETWORK COMMUNICATIONS INTERFACE (QDNC)

The QDNC subsystem appears in the middle of Figure B-1. This subsystem deals with 1) routing of queries to data resources, both local and external, 2) tracking and logging of incoming and outgoing network communications, and 3) the initiation of transmission of data from local to external nodes.

a. Query Distribution Executive

This module tracks and routes queries and responses. For incoming queries, it makes an entry in the incoming query queue with status of "awaiting processing." The Distribution Executive polls the outgoing query queue for entries with "response received" status. Upon finding one, the Distribution Executive notifies the report and display function passing the message sequence number of the response file.

b. Network Communications Interface

The Network Communications Interface initiates transmission of data from the local node to any other network node and receives data into the local node from other netowrk nodes. The data which will pass through the Interface will consist of:

- o Queries being distributed to other network nodes for processing
- o Incoming responses from distributed queries
- o Queries coming into the node from other network nodes
- o Outgoing responses to queries from other nodes
- o Bulletins from one network to all other nodes to announce that a NAD update has taken place at the originating node

- A request from one network node to a second node for transmission of the second node's NAD segment

- A NAD segment transmitted from one network node to another.

Incoming and outgoing messages are blocked and routed using the communications modules of TISS. An outgoing "message" (any data transmissions between network nodes), is converted to a WCF (WICS Common Format) header block including the message sequence number (MSN), user identification, node identifier, message type (query, query response, NAD bulletin, etc.) Outgoing message traffic is handled by polling two queues: the outgoing query queue (for queries to be sent), and the incoming query queue (for responses to be sent). Incoming message traffic is also handled, with most message types being passed immediately to other network elements for processing.

6. RESPONSE NORMALIZATION (RN)

The RN subsystem appears in the bottom portion of Figure B-1. The Response Normalizer, in conjunction with the Response Normal Format (RNF), is designed to provide a capability for output standardization in the overall TIIN design. In general, the task of the RN is to provide the capability for converting data to a standard format after it has been retrieved from a host DBMS. Since data which is returned in response to a single query may be from separate files, it is likely to exhibit different characteristics with regard to coding conventions. The RN normalizes responses, converting them to a standard format which enables collation and reconciliation of conflicts and duplications. The RN also facilitates storage of data in the analyst's private file where it would be available for further processing (such as refining the original request, sorting, statistical analysis, graphic display, etc.).

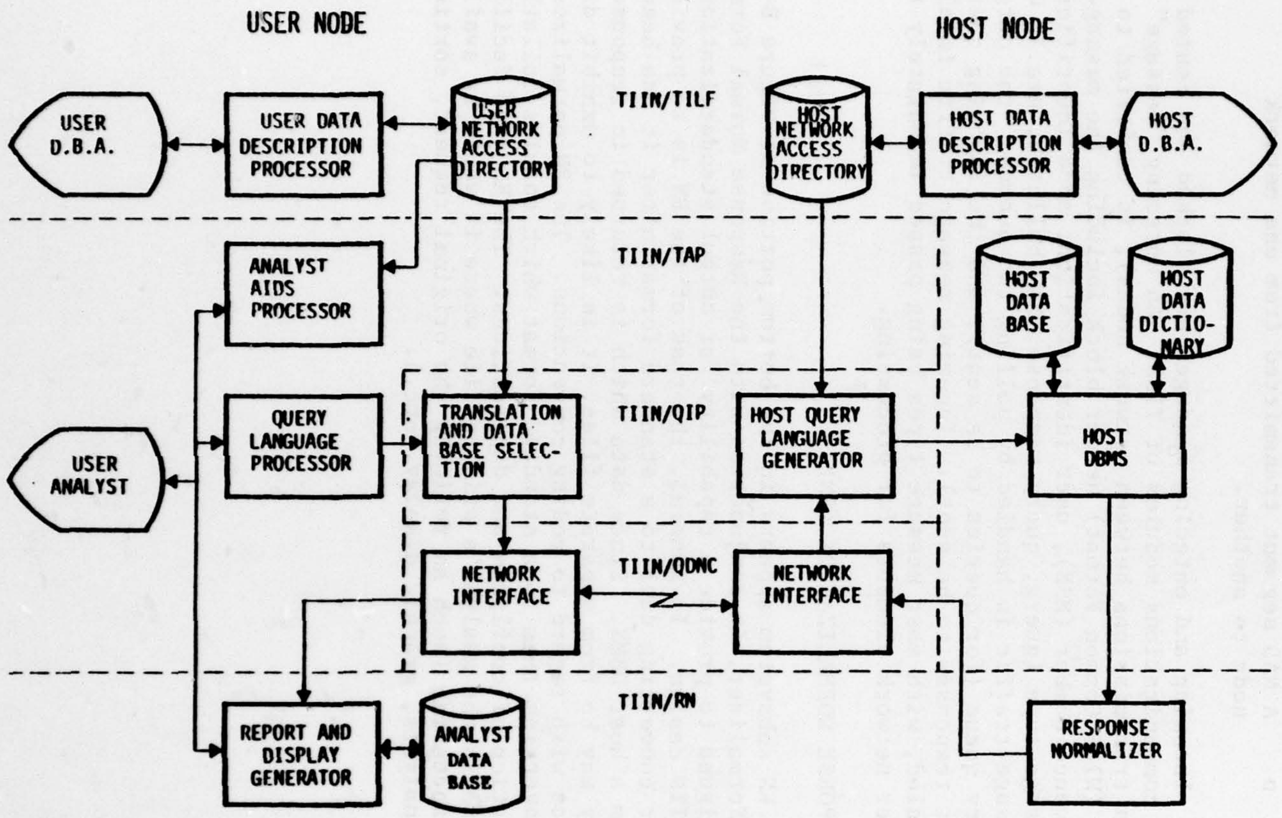


Figure B-1. Transparent Integrated Intelligence Network (TIIN) Modules

APPENDIX C

BIBLIOGRAPHY

1. DATA BASES

Aschim, Frode, "Data Base Networks - An Overview," Management Informatics, Vol. 3, No. , 1974.

Bleier, Robert E., "Treating Hierarchical Data Structures in the SDC Time-Shared Data Management System (TDMS)," 1967 ACM National Meeting, pp. 41-49.

Booth, Grayce M., Honeywell Information Systems, Phoenix, Arizona, "The Use of Distributed Data Bases in Information Networks," First International Conference on Computer Communication: Impacts and Implications, October 24-26, 1972.

Boyce, R.F., Chamberlin, D.D., King III, W.F., and Hammer, M.M., "Specifying Queries as Relational Expressions: SQUARE," IBM Technical Report RJ 1291, October 1973.

Chamberlin, D.D. and Boyce, R.F., "SEQUEL: A Structured English Query Language," Proc. 1974 ACM SIGFIDET Workshop, Ann Arbor, Michigan, (April 1974), pp. 249-264.

Chandra A.N., "Some Considerations in the Design of Homogeneous Distribution Data Bases," IBM Thomas J. Watson Research Center, Yorktown Heights, New York.

CODASYL Development Committee, "An Information Algebra," Communications of the ACM, Vol. 4, (April 1962), pp. 190-204.

CODASYL Data Base Conversion Task Group, Final Report, Information Processing, Vol. 2, No. 1, January 1977.

CODASYL Systems Committee, "Feature Analysis of Generalized Data Base Management Systems," New York, May 1971.

CODASYL Data Base Task Group Report, ACM April 1971.

Codd, E.F., "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM, Vol. 13, No. 6, (June 1970), pp. 377-387.

Codd, E.F., "Seven Steps to Rendezvous with the Casual User," IFIT TC-2 Working Conference on Data Base Management Systems, Cargese, Corsica, 1-5 April 1974.

Computing Surveys, ACM, Vol. 8, No. 1, March 1976.

Date, C.J. "An Architecture for High-Level Language Data Base Extensions," 1976 SIGMOD Conference.

Date, C.J., and Hopewell, P., "File Definition and Logical Data Independence," Proceedings of the 1971 ACM SIGFIDET Workshop.

Defense Intelligence Agency Regulation 65, Intelligence Information Systems, Defense Intelligence Data Standards System (IDSS).

Earley, Jay, "Toward an Understanding of Data Structures," Comm. of the ACM Vol. 14, No. 10. October 1971.

Fredericksen, D.H., "Describing Data in a General Purpose Computer Network," IBM Research Report RC 4122, November 1972.

Fry, J.P. "Distributed Data Bases: A Summary of Research," The University of Michigan, Data Translation Project Working Paper, DE 801, August 1975.

Fry, J.P., Smith, D.P., and Taylor, R.W., "An Approach to Stored Data Definition and Translation," Proc. ACM SIGFIDET Workshop on Data Definition and Access, Denver, Colo., (1972), pp. 13-55.

Fry, J.P., "Stored Data Definition and Translation Approach to the Data Portability Problem," Data Translation Project Report, University of Michigan, Ann Arbor, Mich., February 1974.

Ghosh, S.P., and Senko, M.E., "String Path Search Procedures for Data Base Systems," IBM Journal of Research and Development, Vol. 18, No. 5, (September 1974), pp. 408-422.

Jervis, B., Parker, J., "An Approach for a Working Relational Data System," Dept. of Computer Science, University of British Columbia, Vancouver, Canada.

Logicon, Inc., "ADAPT I Uniform Data Language (UDL) A Preliminary Specification," (July 1976) San Diego, CA.

Logicon, Inc., Final Report - Study of the Multi-Language Problem in COINS: Vol. 1. - Recommended Solutions, Vol. 2. - COINS DBMS Feature Analysis, Vol. 3. - Functional Specification. (May 1975) San Diego, CA.

Lum, V.Y., Shu, N.C., Housel, B.C., "Data Translation, Part I: A General Methodology for Data Conversion and Restructuring," IBM (San Jose), RJ 1525 (#23112), July 1975.

Marcus, Richard S., "A Translating Computer Interface for a Network of Heterogeneous Interactive Information Retrieval Systems," MIT, Electronic Systems Laboratory, 1972.

Merten, A.G., and Fry, J.P., "A Data Description Language Approach to File Translation," ACM Proc. SIGMOD Workshop on Data Description, Access and Control, Ann Arbor, Mich., (1974), pp. 191-205.

Plagman, G.K., and Altshuler, G.P., "A Data Dictionary/Directory System Within the Context of an Integrated Corporate Data Base," AFIPS, 1972, FJCC.

Rosenthal, R., "A Review of Network Access Techniques with a Case Study: The Network Access Machine," National Bureau of Standards Technical Note 917, July 1976.

Schneider, L.S., "A Relational View of the DIAM," Proceedings 1976 ACM SIGMOD International Conference on Management of Data, Washington, D.C., (June 1976), pp 75-90.

Senko, M.E., Altman, E.B., Astrahan, M.M., and Fehder, P.L., "Data Structures and Accessing in Data-Base Systems," IBM Systems Journal, Vol. 12, No. 1, (1973), pp. 30-93.

Shoshani, Ari, "Data Sharing in Computer Networks," 1972 Wescon Tech. Papers, System Development Corp.

Shoshani, Ari, and Spiegler, I., "The Integration of Data Management Systems on a Computer Network," American Institute of Aeronautics and Astronautics, Computer Network Systems Conference, Huntsville, Ala., April 1973.

Shoshani, Ari, and Brandon, K., "The Implementation of a Logical Data Base Converter," October 1975, System Development Corporation, TM-5590/000/00.

Shu, N.C., Housel, B.C., and Lum, V.Y., "Data Translation, Part III. CONVERT: A High Level Translation Definition Language for Data Conversion." IBM Res. Rep. RJ 1515, February 1975.

Smith, Diane P., "An Approach to Data Description and Conversion," PhD dissertation, University of Pennsylvania, Philadelphia, 1971.

Taylor, R.W. "Generalized Data Base Management System Data Structures and their Mapping to Physical Storage," PhD dissertation, University of Michigan, Ann Arbor, 1971, University Microfilms 72-15014.

2. INCO PUBLICATIONS

Development Planning Factors for a Transparent Integrated Intelligence Network, INCO, INC., 1974.

Russek, Marianne, Distributed Data Base Information Systems Technology, INCO, INC., McLean, Virginia 1975.

The Transparent Intelligence Language Facility, Final Technical Report, INCO, INC., McLean, Virginia, June 1976.

Transparent Integrated Intelligence Network Query Intermediate Processor, INCO, INC., McLean, Virginia, January 1977, RADC-TR-77-39, AD-A037 947/9WC.

TIIN Terminal Access Procedures Final Technical Report, INCO, INC., McLean, Virginia, January 1978.

3. LANGUAGE MANUALS

DIAOLS ADP User's Guide for On-Line and Remote/Local Batch Operations, DIA, Washington, D.C., 1974.

On-Line System Support Center (OSSC) User's Guide to DIAOLS/COINS, (to be published).

SEAWATCH External User's Manual, NOSIC, 1974.

TIPS Interrogation Language (TILE) Training Manual for NSA TIPS and COINS Users, NSA, 1972.

TOSS Information Management System (TIMS) On-Line User's Manual, July 1975, INCO, INC., McLean, Virginia.

MISSION

of

Rome Air Development Center

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C^3) activities, and in the C^3 areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

AMERICAN REVOLUTION BICENTENNIAL 1776-1976